University of Thessaly
School of Engineering
Department of Mechanical Engineering

Diploma Thesis

On the development of a 9 – node 2D 2^{nd} order Lagrangian finite element with linear pressure for metal plasticity

by

ANASTASIA TZOUMAKA

Submitted in Partial Fulfilment of the
Requirements for the Diploma in Mechanical Engineering
2016

Approved by the Following Members of the Advisory Committee:

1^{st} **Member (Supervisor)**

Nikolaos Aravas
Professor of Computational Mechanics,
Department of Mechanical Engineering, University of Thessaly

2^{nd} **Member**

Gregory Haidemenopoulos
Professor of Physical Metallurgy,
Department of Mechanical Engineering, University of Thessaly

3^{rd} **Member**

Alexis Kermanidis
Assistant Professor of Mechanical Behavior of Materials,
Department of Mechanical Engineering, University of Thessaly

Abstract

On the development of a 9 – node 2D 2^{nd} order Lagrangian finite element with linear pressure for metal plasticity

Anastasia Tzoumaka

Supervisor: Professor N.Aravas

This work is concerned with the development of a 9 – node 2D and 2^{nd} order Lagrangian finite element with linear pressure for metal plasticity. The development of this 2^{nd} order finite element is based on the theory of finite element method along with finite deformation metal plasticity. Motivated by the increased accuracy associated with this element due to the existence of the 9^{th} node in the middle of the element along with the fact that the Abaqus [2] element library does not include 9 – node elements, we gradually build the methodology that has to be followed to develop this element. The development of this finite element implemented in the Abaqus general purpose finite element software in the form of a User ELement subroutine, while the constitutive model for finite deformation metal plasticity is implemented in the form of a User MATerial subroutine within UEL. The User ELement subroutine is rigorously tested against a series of one element tests using uniaxial tension and compression using either displacement or force control. Through the FEM simulations we verified that UEL works properly on these cases of loading. Finally, we present the problem of wire drawing but using only Abaqus axisymetric elements as the user defined elements had several difficulties in their implementation into such a complex and involved problem.

Keywords: User defined Elements; Metal plasticity; 2^{nd} order Lagrangian finite elements;

Acknowledgements

I would like to express my sincere gratitude to my diploma thesis advisor, Professor Nikolaos Aravas for introducing me to solid and computational mechanics, for instilling me with his knowledge, for the all the valuable advice and continuous guidance throughout my studies. His expertise and personality will always be a form of inspiration for me.

Also, I am deeply thankful to the members of the advisory committee Professor Gregory Haidemenopoulos and Assistant Professor Alexis Kermanidis for all the valuable time they devoted in reviewing this thesis but also for their numerous insightful comments during my undergraduate studies.

I especially need to thank Nick for the undivided moral support, and for always being by my side all these years. Without him, I would not be able to endure all the difficulties, nor achieve so much.

Finally and more importantly, I express my deepest gratitude to my parents and my siblings for all their love and support, without whom I would not be able to complete my studies.

Contents

List of Tables

List of Figures

Introduction

Nonlinear problems in the field of solid and continuum mechanics are often the most exciting but also, the ones that are the most difficult to solve. The finite element method is nowadays well established and is widely used in both academia and industry to accurately solve the most difficult problems. The finite element method is fundamentally based on the concept of discretizing the problem's geometry with finite elements. The solution is then numerically determined within each finite element and the global solution is an interpolation of the local solution within each element.

Commercial finite element software such as Abaqus [1] are equipped with a large library of finite elements that can be used in a variety of problems ranging from 1D to 3D, and are able to model trusses, beams, shells, membranes, plates, and solid bodies. However, in certain applications such as metal forming processes, there is a need for increased solution accuracy which cannot be achieved through mesh refinement. Motivated by this need as well as by the fact that Abaqus provides a general interface through which the implementation of a user-defined element is possible, we formulate a 2 dimensional 9 node lagrangian finite element using the mixed formulation.

This thesis is concerned with the development of a 9 – node 2D and 2^{nd} order Lagrangian finite element with linear pressure for metal plasticity. The 1^{st} Chapter serves as an introduction to Metal Plasticity, presented in the books by Asaro[8], Bigoni[10], Hill[14], Johnson[16], Lubliner[22] and Tomlenov[28]. The 2^{nd} Chapter serves as an introduction to the fundamental principles of Continuum mechanics, presented in the books by Aravas[6], Asaro and Lubarda [8], Bigoni[10], Bonet[11], Anand[13], Hjelmstad[15] and Malvern[23]. The first two chapters, both intend to establish a fundamental theoretical background by presenting principles necessary for the development of the UEL and UMAT subroutines. The 3^{rd} Chapter presents the constitutive model itself giving emphasis in the elastic and plastic equations

governing metal plasticity , as described by Aravas[6], Fionn[12] and Lee[20], and the numerical integration of these elastoplastic equations along with a brief description of Backward and Forward Euler numerical integration schemes. In Chapter 4, we introduce the finite element approximation of the boundary value problem for finite deformations. We present and dicsuss the 2 9 – node Lagrangian finite elements developed in the context of this thesis in the form of UEL subroutines along with their computational implementation. Finally, in Chapter 5 we present and comment on the results of the finite element simulations performed in order to verify that the UEL subroutines developed, are implemented correctly. We test the user elements against several types of problems and compare the results against analytical solutions (if they exist) and the results from Abaqus elements. In particular, we conducted a series of one element tests with uniaxial tension and compression using both force and displacement control as well as contact interactions.

Standard notation is assumed throughout. Fraktur symbols like $\mathfrak{B}$, denote body configurations whereas Ralph Smith's Formal script symbols like $\mathscr{A}$ denote sets. Boldface symbols denote tensors the orders of which are indicated by the context[1]. All tensor components are written with respect to a fixed Cartesian coordinate system with base vectors $\mathbf{e}_i$ $(i = 1, 2, 3)$, and the summation convention is used for repeated Latin indices, unless otherwise indicated. The prefix det indicates the determinant, a superscript T the transpose, a superposed dot the material time derivative, and the subscripts s and a the symmetric and anti-symmetric parts of a second order tensor. Let $\mathbf{a}$, $\mathbf{b}$ be vectors, $\mathbf{A}$, $\mathbf{B}$ second-order tensors, and $\boldsymbol{\mathcal{C}}$ a fourth-order tensor; the following products are used in the text $\mathbf{a}\cdot\mathbf{b} = a_i\, b_i$, $(\mathbf{a}\,\mathbf{b})_{ij} = a_i\, b_j$, $(\mathbf{a}\,\mathbf{b}\,\mathbf{c}\,\mathbf{d})_{ijkl} = a_i\, b_j\, c_k\, d_l$, $(\mathbf{A}\cdot\mathbf{a})_i = A_{ik}\, a_k$, $(\mathbf{a}\cdot\mathbf{A})_i = a_k\, A_{ki}$, $\mathbf{A} : \mathbf{B} = A_{ij}\, B_{ij}$, $(\mathbf{A}\cdot\mathbf{B})_{ij} = A_{ik}\, B_{kj}$, $(\mathbf{A}\,\mathbf{B})_{ijkl} = A_{ij}\, B_{kl}$, $\mathbf{a}\cdot\mathbf{A}\cdot\mathbf{b} = a_i\, A_{ij}\, b_j = (\mathbf{a}\mathbf{b}) : \mathbf{A}$, $(\boldsymbol{\mathcal{C}} : \mathbf{A})_{ij} = \mathcal{C}_{ijkl}\, A_{kl}$, $(\mathbf{A} : \boldsymbol{\mathcal{C}})_{ij} = A_{kl}\, \mathcal{C}_{klij}$, $\mathbf{A} : \boldsymbol{\mathcal{C}} : \mathbf{B} = A_{ij}\, \mathcal{C}_{ijkl}\, B_{kl}$ and $(\mathcal{C} : \boldsymbol{\mathcal{D}})_{ijkl} = \mathcal{C}_{ijpq}\, \mathcal{D}_{pqkl}$. The inverse $\boldsymbol{\mathcal{C}}^{-1}$ of a fourth-order tensor $\boldsymbol{\mathcal{C}}$ that has the 'minor' symmetries $\mathcal{C}_{ijkl} = \mathcal{C}_{jikl} = \mathcal{C}_{ijlk}$ is defined so that $\boldsymbol{\mathcal{C}} : \boldsymbol{\mathcal{C}}^{-1} = \boldsymbol{\mathcal{C}}^{-1} : \boldsymbol{\mathcal{C}} = \boldsymbol{\mathcal{I}}$, where $\boldsymbol{\mathcal{I}}$ is the symmetric fourth-order identity tensor with Cartesian components $\mathcal{I}_{ijkl} = (\delta_{ik}\,\delta_{jl} + \delta_{il}\,\delta_{jk})/2$, δ_{ij} being the Kronecker delta.

[1] In an effort to be as consistent as possible, capital boldface symbols are used for 2nd order tensors whereas lowercase boldface symbols are used to represent vectors. Cauchy and Kirchhoff stress measures are the only exceptions, which although they represent 2nd order tensors, are denoted as $\boldsymbol{\sigma}$ and $\boldsymbol{\tau}$ respectively

Chapter 1

Elementary Continuum Plasticity

1.1 Decomposition of strain

Let us consider a uniaxial tensile test with its stress-strain curve shown in the figure below. Plasticity occurs when stress equals to the uniaxial stress σ_y after which hardening commences. Now, if we choose to unload at a strain of ε until stress reaches zero, we can discern among two components of strain. The two components of ε are the plastic strain ε^p which is the strain remaining when stress equals to zero after the unloading and the elastic strain ε^e which is the recovered strain.

It can be shown [22] that the total strain ε is the sum of the plastic strain and the elastic strain:

$$\varepsilon = \varepsilon^{el} + \varepsilon^{pl} \tag{1.1}$$

which is called classical additive decomposition of strain. In addition, it is obvious from the figure above (1.1) that the stress achieved at a strain of ε is given by:

$$\sigma = E\varepsilon^{el} = E\left(\varepsilon - \varepsilon^{pl}\right) \tag{1.2}$$

We conclude by referring that during materials processing $\varepsilon^e \approx 0$ so that $\varepsilon^{pl} = \varepsilon$ as the elastic strains achieved are very small compared to the plastic one.

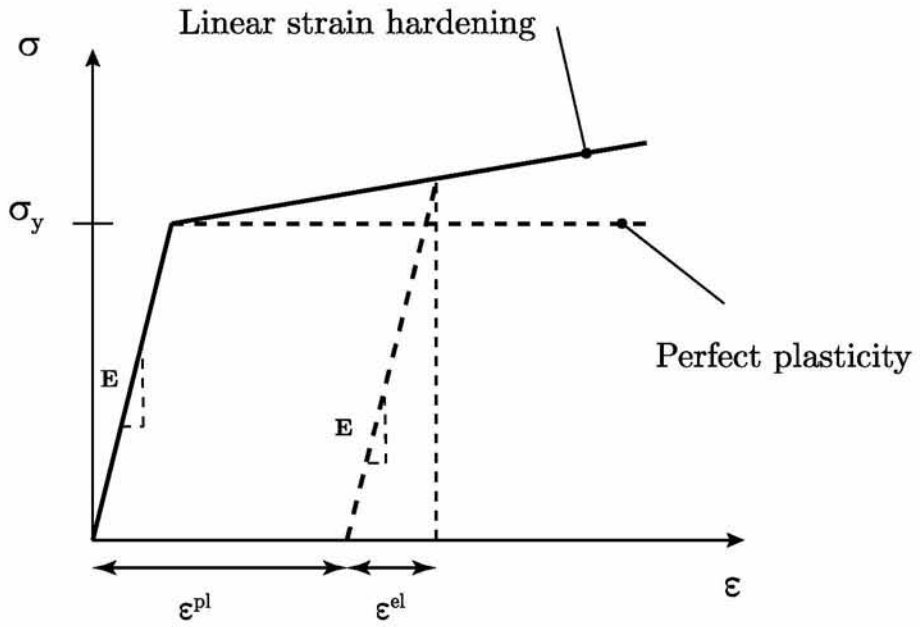


Fig. 1.1: Illustration of a stress-strain curve in a uniaxial tensile test with linear hardening. It is shown clearly the elastic and the plastic part of strain.

1.1.i Incompressibility condition

When plastic deformation takes place we notice that there is not volume change. This condition is called as *Incompressibility condition*. In mathematical form this means that the sum of the plastic strain rate components are zero:

$$\dot{\varepsilon}_x^{pl} + \dot{\varepsilon}_y^{pl} + \dot{\varepsilon}_z^{pl} = 0 \tag{1.3}$$

which easily can be shown by considering a cube, with x, y and z as dimensions, which undergoes plastic uniform deformation.

The fact that the volume remains constant requires that:

$$xyz = x_0 y_0 z_0$$

and by differentiating both sides with respect to time and dividing by xyz we get the incompressibility condition:

$$\frac{\dot{x}}{x} + \frac{\dot{y}}{y} + \frac{\dot{z}}{z} = 0, \quad \dot{\varepsilon}_x = \frac{\dot{x}}{x} \quad \left(\text{recall: } \varepsilon_x = \ln\left(\frac{x}{x_0}\right)\right) \tag{1.4}$$

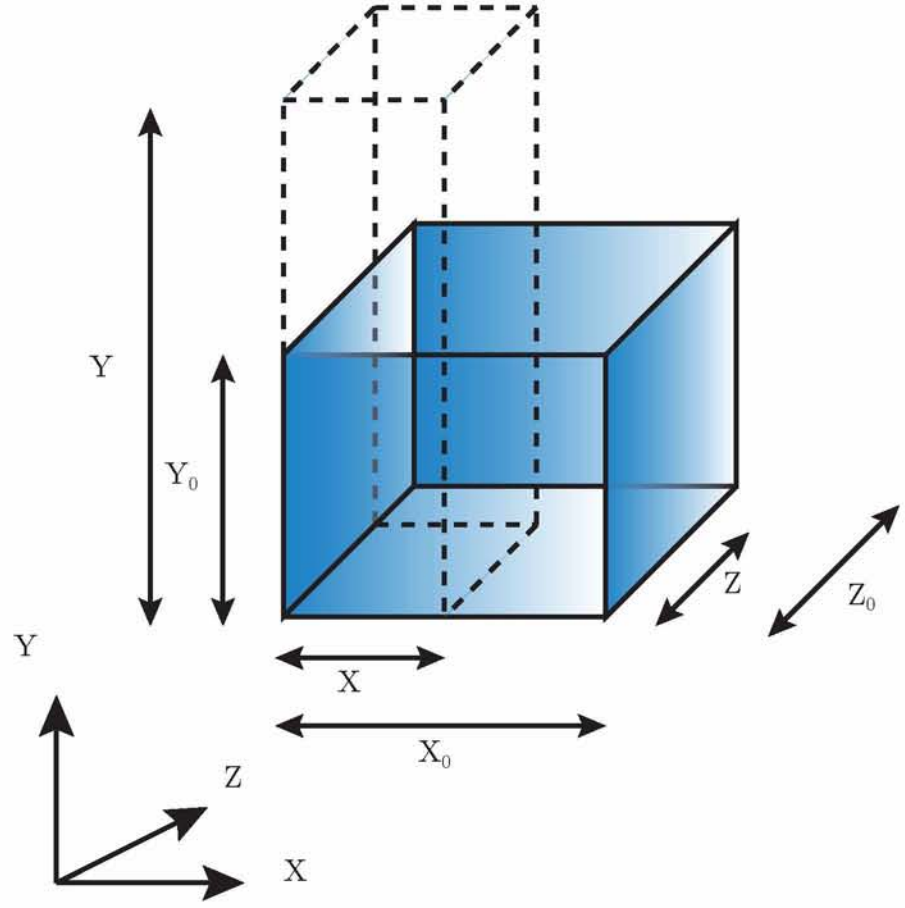


Fig. 1.2: Illustration of a cube undergoing plastic, incompressible, elongation in Y – direction.

1.2 Yield criteria

1.2.i Yield criterion - Yield surface

In the case of uniaxial tensile loading the yield condition is straightforward and is given by:

- $\sigma < \sigma_y \Rightarrow$ elastic behaviour
- $\sigma \geq \sigma_y \Rightarrow$ plastic flow

If we consider a cartesian system with the uniaxial tensile loading along with the ξ_1 axis, then the plastic yield criterion can be written in the form:

- $\Phi(\sigma_{11}) < 0 \Rightarrow$ elastic behaviour
- $\Phi(\sigma_{11}) = 0 \Rightarrow$ plastic flow

where $\Phi(\sigma_{11}) = \sigma_{11} - \sigma_y$ is the yield function.

However, for a multi axial stress state the yield criterion is not straightforward as more than one direct stress exists. In such a case, we have a three dimensional stress state $\boldsymbol{\sigma} = \sigma_{ij} e_i e_j = \Sigma_{i=1}^{3} \sigma_i \mathbf{n}^{(i)} \mathbf{n}^{(i)}$, where σ_i are the principal stresses and e_i are the principal directions of those stresses. Therefore, we assume that a **Yield function** Φ of all the components σ_{ij} exists and thus the yield criterion can be written in the form:

- $\Phi(\boldsymbol{\sigma}) < 0 \Rightarrow$ elastic behaviour
- $\Phi(\boldsymbol{\sigma}) = 0 \Rightarrow$ plastic flow

Taking into account that the Cauchy stress tensor $\boldsymbol{\sigma}$ is a symmetric tensor [1], we can consider 6 independent components σ_{ij} and thus $\Phi(\boldsymbol{\sigma})$ is a function of 6 variables. For an isotropic material, the yield function depends only on the magnitude of the principal stresses and not from their principal directions. As a result, we can write the following:

$$\Phi(\boldsymbol{\sigma}) = \Phi(\sigma_1, \sigma_2, \sigma_3) = 0 \tag{1.5}$$

or

$$\Phi(I_1, I_2, I_3) = 0 \tag{1.6}$$

where I_1, I_2 and I_3 are the invariants of tensor $\boldsymbol{\sigma}$ and they are defined as:

$$I_1 = \text{trace}(\boldsymbol{\sigma}) = \sigma_{kk}$$
$$I_2 = \frac{1}{2}\left[(\text{trace}(\boldsymbol{\sigma}))^2 - \text{trace}(\boldsymbol{\sigma})^2\right] = \frac{1}{2}\left(I_1 - \sigma_{ij}\sigma_{ij}\right)$$
$$I_3 = det[\boldsymbol{\sigma}] = \frac{1}{6}\left(I_1^3 + 3\sigma_{ik}\sigma_{kl}\sigma_{li} - 3I_1\sigma_{ij}\sigma_{ij}\right)$$

In the case of isotropic non-porous metal materials, the yield criterion is independent of the hydrostatic stress $p = \frac{1}{3}\sigma_{kk} = \frac{1}{3}I_1$ and therefore we have to express the stress tensor $\boldsymbol{\sigma}$ in terms of the stress deviatoric $\boldsymbol{s}$. The stress deviatoric is defined by the following equation:

$$\boxed{\boldsymbol{s} = \boldsymbol{\sigma} - p\boldsymbol{\delta}} \quad \text{or in component form} \quad \sigma_{ij} = \sigma_{ij} - p\delta_{ij} \tag{1.7}$$

Looking at equation (1.7) we can sum up on the following statements:

[1] A proof of the symmetry of the Cauchy stress tensor can be found in every book which is concerned with the mechanics of Solids such as Asaro[8].

- The trace($\boldsymbol{s}$) $= s_{kk} = 0$ and as a result the 1^{st} invariant of $\boldsymbol{s}$ which is denoted as J_1 is always equal to zero ($J_1 = s_{kk} = 0$), and
- The shear components of $\boldsymbol{\sigma}$ are equal to the shear components of $\boldsymbol{s}$

$$s_{ij} = \begin{cases} \sigma_{ij}, & \text{if } i \neq j \\ \sigma_{ij} - p, & \text{if } i = j \end{cases}$$

Now, taking into account the above statements we can write the yield function for an isotropic non-porous metal material as follows:

$$\Phi(J_2, J_3) = 0 \tag{1.8}$$

where J_2 and J_3 are quantities which are related to the stress deviatoric's invariants through

$$J_2 = -I_2^{dev} = \frac{1}{2} s_{ik} s_{ki} \geq 0 \quad \text{and} \quad J_3 = I_3^{dev} = \det[\boldsymbol{\sigma}] = \frac{1}{3} s_{ik} s_{kl} s_{li}$$

1.2.ii The Π-plane

Let us consider an equation which defines a plane:

$$\sigma_1 + \sigma_2 + \sigma_3 = 0 \tag{1.9}$$

where σ_1, σ_2 and σ_3 are the principal stresses. This plane, is known as the Π-*plane* and it is very useful for the definition of the Yield criteria. The normal to the Π−plane unit vector is:

$$\mathbf{n} = \frac{1}{\sqrt{3}} (\mathbf{e}_1 + \mathbf{e}_2 + \mathbf{e}_3) \tag{1.10}$$

where $(\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3)$ are the unit vectors on the space of principal axis.

Now, we consider an arbitrary point S on the space of the principal stresses which corresponds to the stress state $(\sigma_1, \sigma_2, \sigma_3)$. The position vector of the aforementioned point, S, is the $\vec{OS}$

$$\vec{OS} = \sigma_1 \mathbf{e}_1 + \sigma_2 \mathbf{e}_2 + \sigma_3 \mathbf{e}_3 \tag{1.11}$$

The vector $\vec{OS}$ can be analyzed in two components $\vec{OP}$ and $\vec{PS}$ and we can summarize on the following statements:

- The projection $\vec{OP}$ of $\vec{OS}$ is the deviator part of the stress tensor $\boldsymbol{s}$ and the normal component $\vec{PS}$ is the hydrostatic part of the stress tensor $\boldsymbol{\sigma}$

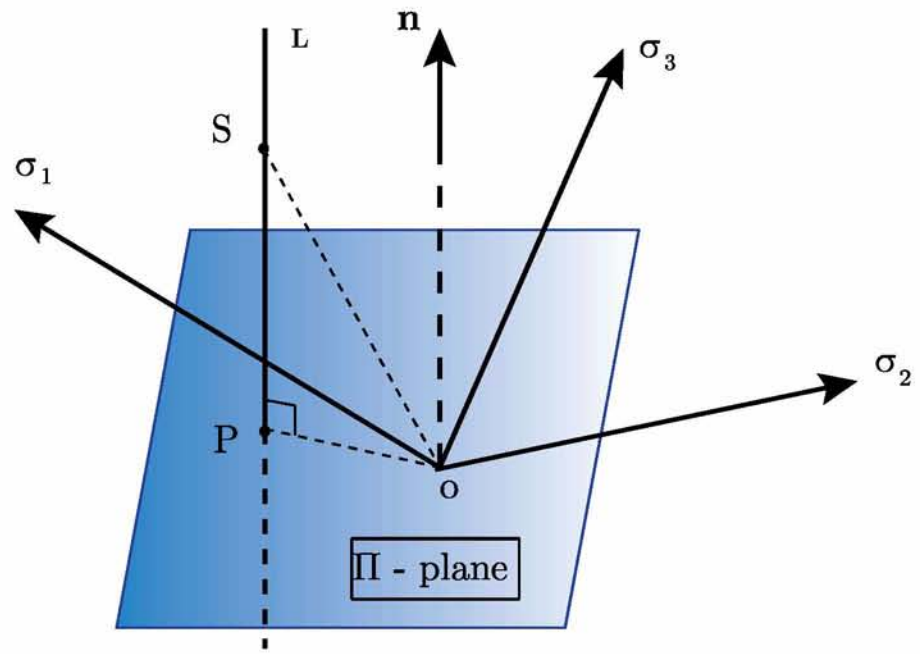


Fig. 1.3: Illustration of The Π *- plane.*

Now, let us consider the following stress condition $(\sigma_1, \sigma_2, \sigma_3)$ which corresponds to the point S on the space of the principal axes. We can define line L as the line on which all points have the same deviatoric part $\boldsymbol{s}$ and different hydrostatic part $p = \frac{\sigma_{kk}}{3}$. Hence, if the stress condition on point S satisfies (1.8) then every point on line L satisfies the yield criterion (1.8). This means that the yield surface is a cylindrical surface with its generators perpendicular to the Π-plane extending to infinity in both directions.

1.2.iii Tresca yield criterion

The Tresca yield criterion assumes that plastic deformation occurs when the maximum shear stress over all planes attains a critical value, denoted as c_1. Then the yield function can be written in the form:

$$\Phi(\boldsymbol{\sigma}) = \frac{1}{2}\max\left(|\sigma_1 - \sigma_2|, |\sigma_2 - \sigma_3|, |\sigma_3 - \sigma_1|\right) - c_1 = 0 \tag{1.12}$$

The projection of the Tresca yield surface in Π-plane is a regular hexagon as shown in the figure below.

It is useful now to determine the relationship between σ_0 and τ_0. For this reason let as consider the uniaxial tensile stress state in which $\boldsymbol{\sigma} = \sigma \mathbf{e}_1\mathbf{e}_1$. In such a case we can easily define that $\sigma_{max} = \sigma$ and $\sigma_{min} = 0$. Plasticity occurs when $\sigma_{max} = \sigma_0$, where σ_0 is the yield stress in a uniaxial tensile stress state, and thus the yield criterion on equation (1.12) becomes:

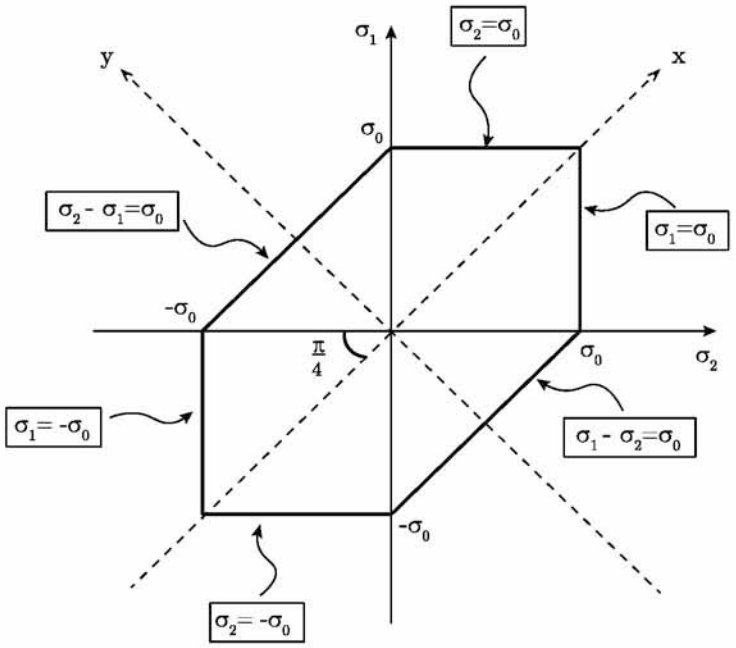


Fig. 1.4: Illustration of Tresca's yield surface in plain stress state for various angles

$$\frac{\sigma_0 - 0}{2} = c_1 \quad \Rightarrow \quad c_1 = \frac{\sigma_0}{2} \tag{1.13}$$

Now let as consider the case of pure shear in which $\boldsymbol{\sigma} = \tau\,(\mathbf{e}_1\mathbf{e}_2 + \mathbf{e}_2\mathbf{e}_1)$. According to Mohr's circle we can define that $\sigma_{max} = \tau$ and $\sigma_{min} = -\tau$. Plasticity occurs when the shear stress τ will be equal to τ_0, where τ_0 is the yield stress in a pure shear stress state. Therefore, the yield criterion on equation (1.12) becomes:

$$\frac{\tau_0 - (-\tau_0)}{2} = c_1 \quad \Rightarrow \quad c_1 = \tau_0 \tag{1.14}$$

Consequently, from equations (1.13) and (1.14) we can easily define the relationship between σ_0 and τ_0:

$$\boxed{\tau_0 = \frac{\sigma_0}{2}} \tag{1.15}$$

1.2.iv Von Mises yield criterion

According to Von Mises, plastic yielding occurs when:

$$\Phi(\boldsymbol{\sigma}) = J_2 - c_2 \tag{1.16}$$

where c_2 is a constant which characterizes the material and J_2 is equal to:

$$J_2 = \frac{1}{6}\left[(\sigma_{11}-\sigma_{22})^2 + (\sigma_{11}-\sigma_{33})^2 + (\sigma_{22}-\sigma_{33})^2\right] + \sigma_{12}^2 + \sigma_{13}^2 + \sigma_{23}^2 \tag{1.17}$$

or

$$J_2 = \frac{1}{2}s_{ij}s_{ij} = \frac{1}{2}\sum_{i=1}^{3}\sum_{j=1}^{3} s_{ij}^2 \tag{1.18}$$

The projection of the Von Mises yield surface in the Π-plane is a 30^o degrees arc with radius $r = \sqrt{2J_2} = \sqrt{2c_2} = \sqrt{\frac{2}{3}}\sigma_0$.

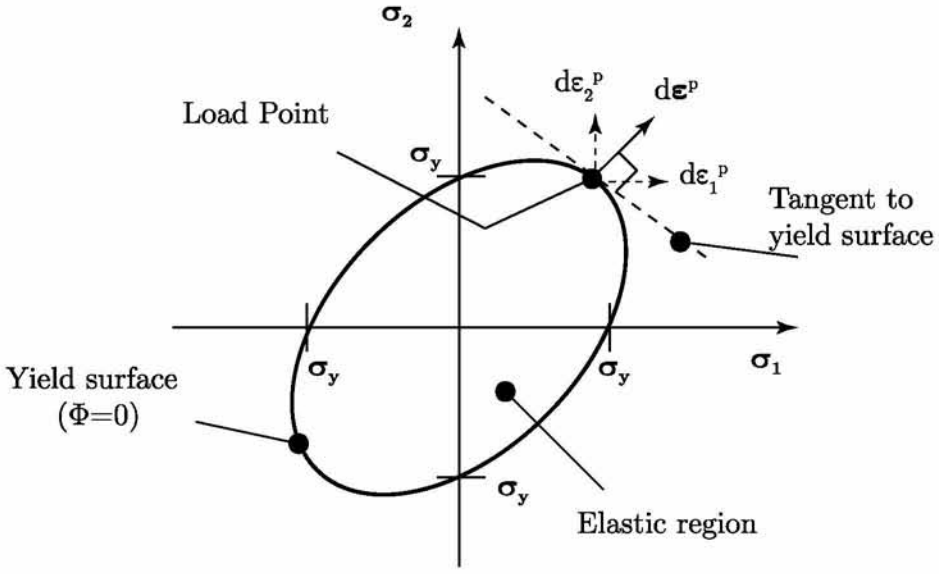


Fig. 1.5: Illustration of the Von Mises yield surface and the rate of plastic strain in the case of plain stress state.

Now, we need to define the relationship between σ_0 and τ_0 as we did previously on the Tresca yield criterion. For this reason we consider again the uniaxial tensile stress state $\boldsymbol{\sigma} = \sigma \mathbf{e}_1\mathbf{e}_2$. Then, we calculate the final form of the Von Mises yield criterion in equation (1.17). It easily can be shown that $J_2 = \frac{\sigma^2}{3}$. Knowing that plastic yielding occurs when $\sigma = \sigma_0$ we conclude on the following equation:

$$c_2 = \frac{\sigma_0^2}{3} \tag{1.19}$$

We consider again the pure shear stress state in which the stress tensor has the form $\boldsymbol{\sigma} = \tau(\mathbf{e}_1\mathbf{e}_1 + \mathbf{e}_2\mathbf{e}_1)$ and thus $J_2 = \tau^2$. Plasticity occurs when the shear stress τ will be equal to the yield stress in a shear stress state τ_0. Therefore, according to the Von mises yield criterion on equation (1.17) the constant c_2 will be equal to:

$$c_2 = \tau_0^2 \tag{1.20}$$

Comparing equations (1.19) and (1.20) we can conclude on the following relationship among σ_0 and τ_0:

$$\tau_0 = \frac{\sigma_0}{\sqrt{3}} \quad \Rightarrow \quad \boxed{\tau_0 = 0.577\sigma_0} \tag{1.21}$$

1.3 Von Mises Plasticity

So far we have discussed about the necessary conditions to initiate yielding. Another important issue is the 'direction' of (plastic) flow, after yielding. In order to determine the direction of flow we introduce the normality rule of plasticity which is given by:

$$d\boldsymbol{\varepsilon}^p = d\lambda \frac{\partial \Phi}{\partial \boldsymbol{\sigma}} \quad \text{or} \quad \dot{\boldsymbol{\varepsilon}}^p = \dot{\lambda} \frac{\partial \Phi}{\partial \boldsymbol{\sigma}} \tag{1.22}$$

The term $\frac{\partial \Phi}{\partial \boldsymbol{\sigma}}$ (where $\boldsymbol{\sigma}$ is a vector) is the direction of the plastic strain increment or the plastic strain rate, while the term $\dot{\lambda}$, which is called plastic multiplier, gives the magnitude of the plastic strain rate.

If we rewrite equation (1.22) in terms of deviatoric stress we take the following expression:

$$d\boldsymbol{\varepsilon}^p = \frac{3}{2} d\lambda \frac{\boldsymbol{s}}{\sigma_{eq}} \tag{1.23}$$

We can easily demonstrate (Lubliner[22]) that for a Von Mises material the plastic multiplier $d\lambda$ is equal to dp or $\dot{p} = \dot{\lambda}$, where:

$$\dot{p} = \left(\frac{2}{3} \dot{\boldsymbol{\varepsilon}}^p : \dot{\boldsymbol{\varepsilon}}^p \right)^{\frac{1}{2}} \qquad \text{and} \qquad dp = \left(\frac{2}{3} d\boldsymbol{\varepsilon}^p : d\boldsymbol{\varepsilon}^p \right)^{\frac{1}{2}}$$

Knowing that for a metal following the Von Mises plasticity model $dp = d\lambda$, we can rewrite equation (1.23) in the terms of dp:

$$d\boldsymbol{\varepsilon}^p = \frac{3}{2} dp \frac{\boldsymbol{s}}{\sigma_{eq}} \tag{1.24}$$

1.4 Consistency Condition

Let us consider a uniaxial tensile loading as shown in figure 1.1. We assume a perfectly plastic material which means that there is no hardening. The fact that there is no hardening means that further plastic deformation achieved with constant stress equals to σ_y. Requiring the load point to remain on the yield surface during plastic deformation we make a condition which is known as the *consistency condition*. The yield function now takes the form:

$$\Phi(\boldsymbol{\sigma}, p) = \sigma_e - \sigma_y = \boldsymbol{\sigma}_e(\boldsymbol{\sigma}) - \sigma_y(p) = 0 \tag{1.25}$$

For an incremental change in stress and effective plastic strain the consistency condition is written as follows:

$$\Phi(\boldsymbol{\sigma} + d\boldsymbol{\sigma}, p + dp) = 0 \quad \text{and} \quad \Phi(\boldsymbol{\sigma} + d\boldsymbol{\sigma}, p + dp) = \Phi(\boldsymbol{\sigma}, p) + \frac{\partial \Phi}{\partial \boldsymbol{\sigma}} : d\boldsymbol{\sigma} + \frac{\partial \Phi}{\partial p} dp$$

we can derive the following:

$$\frac{\partial \Phi}{\partial \boldsymbol{\sigma}} \cdot d\boldsymbol{\sigma} + \frac{\partial \Phi}{\partial p} dp = 0 \tag{1.26}$$

We use Hooke's law to relate the stress and elastic strains:

$$d\boldsymbol{\sigma} = \boldsymbol{\mathcal{L}} d\boldsymbol{\varepsilon}^e \quad \Rightarrow \quad d\boldsymbol{\sigma} = \boldsymbol{\mathcal{L}}(d\boldsymbol{\varepsilon} - d\boldsymbol{\varepsilon}^p) \tag{1.27}$$

where $\boldsymbol{\mathcal{L}}$ is the elastic stiffness matrix. Using now the normality rule, which is given by equation (1.22), we can rewrite the Hooke's law in the following form:

$$d\boldsymbol{\sigma} = \boldsymbol{\mathcal{L}} \left(d\boldsymbol{\varepsilon} - d\lambda \frac{\partial \Phi}{\partial \boldsymbol{\sigma}} \right) \tag{1.28}$$

Using equation (1.28) into equation (1.26) we take the following equation which is very useful to determine the plastic multiplier $d\lambda$:

$$\frac{\partial \Phi}{\partial \boldsymbol{\sigma}} \cdot \boldsymbol{\mathcal{L}} \left(d\boldsymbol{\varepsilon} - d\lambda \frac{\partial \Phi}{\partial \boldsymbol{\sigma}} \right) + \frac{\partial \Phi}{\partial p} dp = 0 \tag{1.29}$$

Recalling now equation (1.30) for dp and substituting the normality rule (1.22) on this equation, we have the following expression for dp:

$$dp = \left(\frac{2}{3} d\lambda \frac{\partial \Phi}{\partial \boldsymbol{\sigma}} \cdot d\lambda \frac{\partial \Phi}{\partial \boldsymbol{\sigma}} \right)^{1/2} \tag{1.30}$$

Substituting equation (1.30) into (1.29) we can compute the plastic multiplier $d\lambda$ and finally the stress increment $d\boldsymbol{\sigma}$ in (1.28):

$$d\lambda = \frac{(\partial \Phi / \partial \boldsymbol{\sigma}) \cdot \boldsymbol{\mathcal{L}} d\boldsymbol{\varepsilon}}{(\partial \Phi / \partial \boldsymbol{\sigma}) \cdot \boldsymbol{\mathcal{L}} (\partial \Phi / \partial \boldsymbol{\sigma}) - (\partial \Phi / \partial p)((2/3)(\partial \Phi / \partial \boldsymbol{\sigma}) \cdot (\partial \Phi / \partial \boldsymbol{\sigma}))^{1/2}} \tag{1.31}$$

$$d\boldsymbol{\sigma} = \left(\boldsymbol{\mathcal{L}} - \boldsymbol{\mathcal{L}} \frac{\partial \Phi}{\partial \boldsymbol{\sigma}} \frac{(\partial \Phi / \partial \boldsymbol{\sigma}) \cdot \boldsymbol{\mathcal{L}}}{(\partial \Phi / \partial \boldsymbol{\sigma}) \cdot \boldsymbol{\mathcal{L}} (\partial \Phi / \partial \boldsymbol{\sigma}) - (\partial \Phi / \partial p)((2/3)(\partial \Phi / \partial \boldsymbol{\sigma}) \cdot (\partial \Phi / \partial \boldsymbol{\sigma}))^{1/2}} \right) d\boldsymbol{\varepsilon} \tag{1.32}$$

1.5 Isotropic Hardening

We will now discuss about hardening in metals. The definition '*hardening*' declares that after plastic flow the stress required to cause further plastic deformation need to be increased. As a result the yield surface seems expanded compared to the original (figure 1.6). If this expansion is uniform in all directions in stress space, then the hardening is referred to as *isotropic*.

The amount of expansion is often taken to be a function of plastic strain p and thus the yield function an be written in the form:

$$\Phi(\boldsymbol{\sigma}, p) = \sigma_e - \sigma_y = \boldsymbol{\sigma}_e(\boldsymbol{\sigma}) - \sigma_y(p) = 0 \tag{1.33}$$

where $\sigma_y(p)$ might be of the form: $\sigma_y(p) = \sigma_{y0} + r(p)$

where:

σ_{y0} is the initial yield stress and r(p) is the isotropic hardening function.

Now if we assume linear isotropic hardening we can write the function r(p) as follows:

$$dr(p) = h \cdot dp \tag{1.34}$$

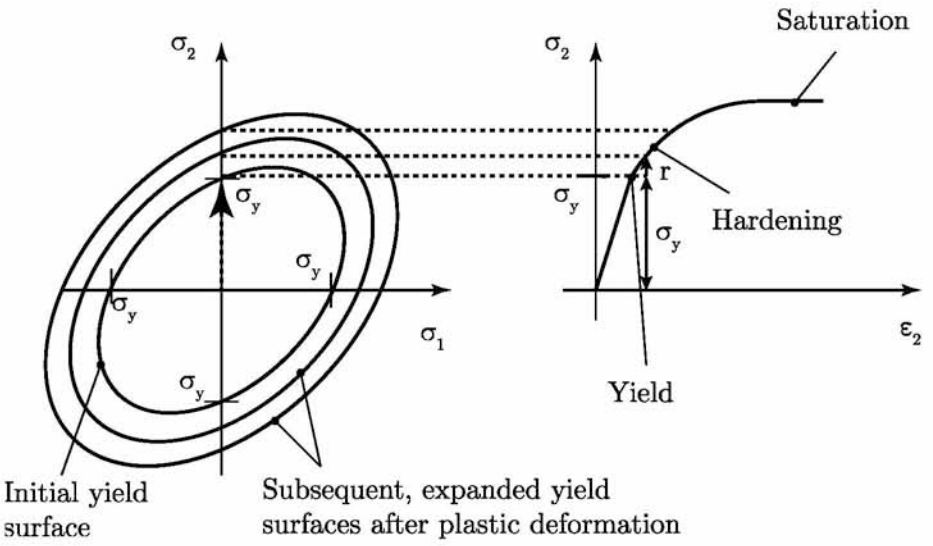


Fig. 1.6: Illustration of the Von Mises yield surface in the case of isotropic hardening.

in which term h is a constant. Taking into consideration equation (1.24), we can conclude that for uniaxial conditions $dp = d\varepsilon^p$. Hence, the stress increase due to isotropic hardening is dr and we can rewrite equation (1.34) as:

$$d\varepsilon^p = \frac{d\sigma}{h} \tag{1.35}$$

Also recall that the increment in elastic strain is:

$$d\varepsilon^e = \frac{d\sigma}{E} \tag{1.36}$$

We know from equation (1.1) that $\varepsilon = \varepsilon^e + \varepsilon^p$. The same equality can be written for the rates as follows:

$$d\varepsilon = d\varepsilon^e + d\varepsilon^p \quad \Rightarrow \quad d\varepsilon = \frac{d\sigma}{E} + \frac{d\sigma}{h} = d\sigma\left(\frac{E+h}{E\cdot h}\right) \Rightarrow \quad d\sigma = E\left(1 - \frac{E}{E+h}\right)d\varepsilon$$

Notice that due to the analysis in section 1.4 about the consistency condition we can write the plastic multiplier ($d\lambda$) in terms of the total strain increment. This form is very important in the development of computational techniques such as the finite element methods. Thus, combining equations (1.26) and (1.30) we obtain:

$$d\lambda = \frac{-(\partial\Phi/\partial\boldsymbol{\sigma})\cdot d\boldsymbol{\sigma}}{(\partial\Phi/\partial p)((2/3)(\partial\Phi/\partial\boldsymbol{\sigma})\cdot(\partial\Phi/\partial\boldsymbol{\sigma}))} \tag{1.37}$$

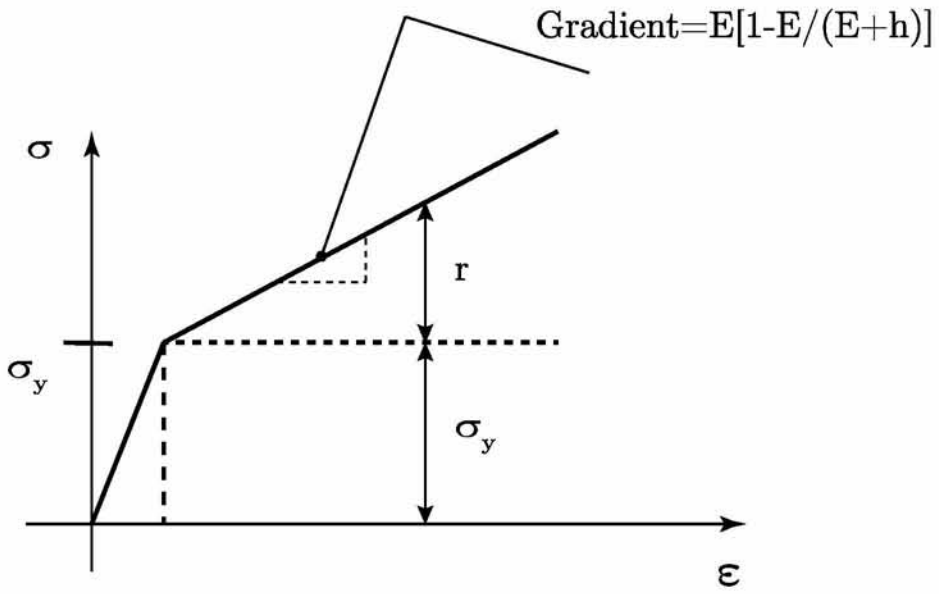


Fig. 1.7: On this figure it is illustrated the $\boldsymbol{\sigma} - \varepsilon$ curve during linear isotropic hardening.

1.6 Work Hardening

Let us consider a deformable body as shown in the figure below. For this deformable body, we can write the equilibrium of mechanical power:

$$\int_V b_i\, \sigma_i\, dV + \int_S t_i\, \sigma_i\, dS = \int_V \sigma_{ij}\, \dot{\varepsilon}_{ij}\, dV \quad \Rightarrow$$

$$P = \underbrace{\int_V \sigma_{ij}\dot{\varepsilon}_{ij}\, dV}_{\text{Stress power}} = \underbrace{\int_V \sigma_{ij}\, d\varepsilon_{ij}^{el}\, dV}_{\text{Total elastic work}} + \underbrace{\int_V \sigma_{ij}\, d\varepsilon_{ij}^{pl}\, dV}_{\text{Total plastic work}} \tag{1.38}$$

The total elastic work can easily be calculated because it depends only from the initial and final condition. If we assume that there is a function $U(\boldsymbol{\varepsilon}^{el})$ for the elastic work, we can write the following:

$$\bar{U} = \sigma_{ij}\, \varepsilon_{ij}^{el} \quad \Rightarrow \quad d\bar{U} = \sigma_{ij}\, d\varepsilon_{ij}^{el} \quad \rightarrow$$

$$\int_{\boldsymbol{\varepsilon}_A^{el}}^{\boldsymbol{\varepsilon}_B^{el}} D\bar{U} = \bar{U}(\boldsymbol{\varepsilon}_B^{el}) - \bar{U}(\boldsymbol{\varepsilon}_A^{el}) \tag{1.39}$$

which essentially means that the total elastic work is independent from the path which is followed during the implementation of body forces. As far as the total plastic work

is concerned, we can not insinuate that it is independent from the path followed during plastic deformation as there are more than one magnitude of strain corresponding to the same magnitude of stress. According to Drucker (Lubliner[22]): *If a unit volume of an elastic-plastic specimen under uniaxial stress is initially at stress* σ *and plastic strain* ε^{pl}*, and if an "external agency" (one that is independent of whatever has produced the current loads) slowly applies an incremental load resulting in a stress increment* $\mathrm{d}\sigma$ *,which causes the elastic and plastic strain increments* $\mathrm{d}\varepsilon^{el}$ and $\mathrm{d}\varepsilon^{pl}$*, respectively, and subsequently slowly removes it, then* $\mathrm{d}\sigma\mathrm{d}\varepsilon{=}\mathrm{d}\sigma(\mathrm{d}\varepsilon^{el}{+}\mathrm{d}\varepsilon^{pl})$ *is the work performed by the external agency in the course of incremental loading, and* $\mathrm{d}\sigma\mathrm{d}\varepsilon^{pl}$ *is the work performed in the course of the cycle consisting of the application and removal of the incremental stress.* (Note that for $\mathrm{d}\varepsilon^{pl} \neq 0$, σ must be the current yield stress). Therefore, from **Drucker's postulate** we can derive the following important statements:

1. A work-hardening plastic material is one in which the work done during incremental loading is positive, and

2. the work done during unloading cycle is nonnegative

The extended definition to general three-dimensional states of stress and strain according to Drucker is:

$$d\sigma_{ij}d\varepsilon_{ij} > 0 \quad \text{and} \quad d\sigma_{ij}d\varepsilon^{pl}_{ij} \geq 0 \tag{1.40}$$

The above inequality which is known as **Drucker's inequality** is valid for both work-hardening and perfectly plastic materials as in the case of perfectly plastic materials equation (1.40) becomes:

$$d\sigma_{ij}d\varepsilon_{ij} \geq 0 \quad \text{and} \quad d\sigma_{ij}d\varepsilon^{pl}_{ij} = 0 \tag{1.41}$$

Now we have to mention that Drucker's statement of his work-hardening postulate can be used even if the additional stress is not a small increment. This concept is illustrated in figure 1.9, where $\boldsymbol{\sigma}^*$ is the initial stress which can be inside the elastic region or at a point on the yield surface far away from $\boldsymbol{\sigma}$. With $\mathrm{d}\boldsymbol{\sigma}$ neglected the Drucker's postulate implies that:

$$(\sigma_{ij} - \sigma^*_{ij})\dot{\varepsilon}^p_{ij} \geq 0 \tag{1.42}$$

The above equation, (1.42), is also called **postulate of maximum plastic dissipation** and is valid for work-softening and perfectly plastic materials. Now we will write Drucker's inequality in vector form in the six-dimensional space:

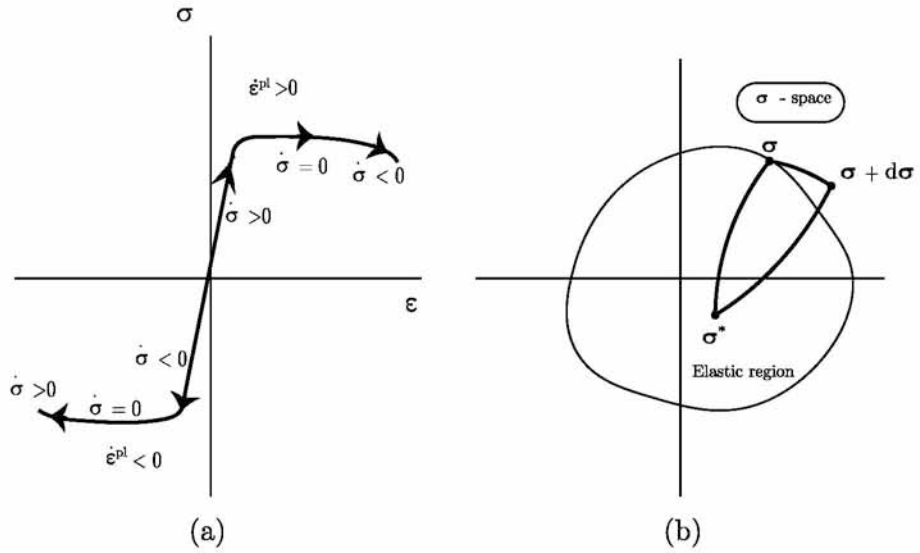


Fig. 1.8: Drucker's postulate: (a) illustration in the uniaxial stress-strain plane; (b) illustration in stress space

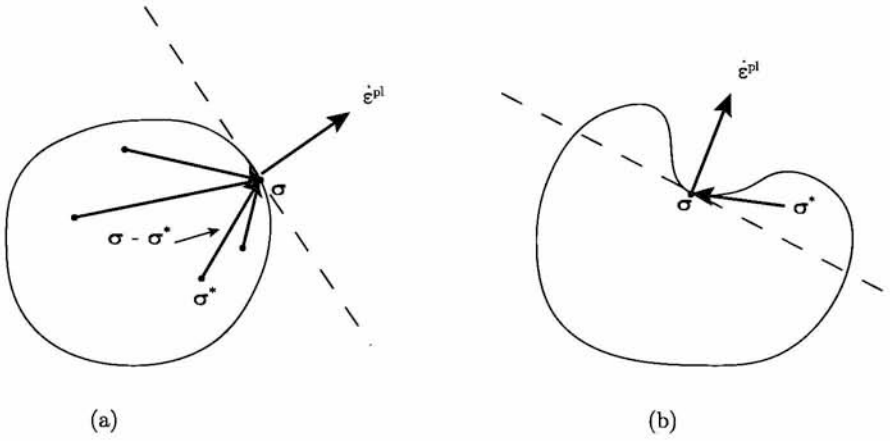


Fig. 1.9: Yield surface: (a) Normality; (b) convexity

$$(\boldsymbol{\sigma} - \boldsymbol{\sigma}^*) \cdot \dot{\varepsilon}^{pl} \geq 0 \tag{1.43}$$

In figure 1.9(a) we consider the yield surface everywhere smooth, so that a well-defined tangent hyperplane and normal direction exists at every point. The Drucker's inequality in equation (1.42) is valid for every $\boldsymbol{\sigma}^*$ to the inward side of the tangent to the yield surface at $\boldsymbol{\sigma}$ if $\dot{\varepsilon}^{pl}$ directed along the outward normal there. This consequence is known as the **normality rule** which we defined in a previously section in equation (1.22). However, as you can see in figure 1.9(b) if there are any $\boldsymbol{\sigma}^*$ lying to the outward side of the tangent, the inequality is violated, thus you can conclude that the yield surface must be always **convex**.

Chapter 2

Continuum mechanics theory

2.1 Description of deformation - Deformation gradient

Let us consider a continuum body which is consisted of an infinite number of material points. For this continuum body, we have to be able to provide an accurate description of changes in the shape, size and orientation. Therefore, we have to define an initial configuration, which we call **reference configuration**, at time $t = t_0$, which we assume to be known and is denoted as $\mathfrak{B}_0$. The choice of a reference configuration is arbitrary. We also define the **deformed configuration** $\mathfrak{B}_t$, as any other configuration of the continuum body at subsequent times t. At this point, **deformation** is defined as any geometry related deviation between the reference and deformed configuration of the body. Next, we represent the analytical expression of the motion of the continuum body by introducing a smooth and differentiable function

$$\mathbf{x} = \mathbf{x}(\mathbf{X}, t): \qquad \text{where: } \mathbf{X} \in \mathfrak{B}_0 \text{ and } \mathbf{x} \in \mathfrak{B}_t \tag{2.1}$$

which satisfies the initial condition: $\mathbf{x}(\mathbf{X}, t_0) = \mathbf{X}$. Now we assume that the function $\mathbf{x} = \mathbf{x}(\mathbf{X}, t)$ is one-to-one which means that existing material points cannot be destroyed as well as new material points cannot be added to the body. Thus, the function has the following mathematical property:

$$\mathbf{x} = \mathbf{x}(\mathbf{X}, t) \;\Leftrightarrow\; \mathbf{X} = \mathbf{X}(\mathbf{x}, t)$$

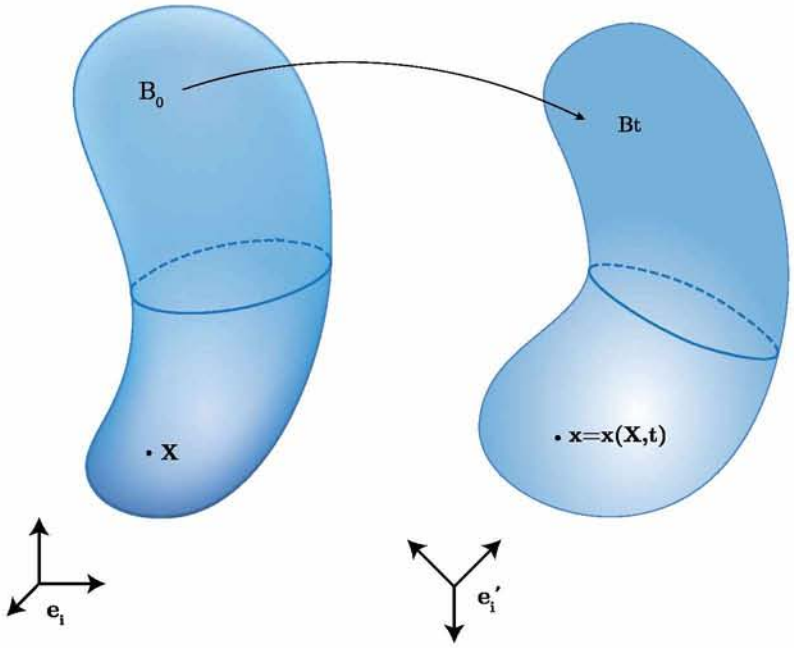


Fig. 2.1: The reference body $\mathfrak{B}_0$ and the deformed body $\mathfrak{B}_t$

If we decide to follow a given material point we set on equation (2.1) $\mathbf{X} = \hat{\mathbf{X}} = ct$, to get:

$$\mathbf{x} = \mathbf{x}(\mathbf{X} = \hat{\mathbf{X}}, t) \;\Rightarrow\; \mathbf{x} = \hat{\mathbf{x}}\,(t)$$

which defines the motion equation for the material point $\hat{\mathbf{X}}$. Assuming that the function $\mathbf{x}(\mathbf{X}, t)$ is twice-differentiable in time, we can define the velocity $\hat{\boldsymbol{v}}(t)$ and acceleration $\hat{\mathbf{a}}(t)$ of material point $\hat{\mathbf{X}}$ as:

$$\hat{\boldsymbol{v}}(t) = \frac{d\hat{\mathbf{x}}(t)}{dt} \qquad \hat{\mathbf{a}}(t) = \frac{d^2\hat{\mathbf{x}}(t)}{dt^2}$$

Those expressions are useful when we are interested in studying the dynamics of a desired material point. However, if we are interested in the description of the kinematics of deformation of the continuum body, the aforementioned expressions are not suitable. In such a case, we use the equation (2.1) at a given time instance $t = \hat{t}$ and we get:

$$\mathbf{x} = \mathbf{x}(\mathbf{X}, t = \hat{t}) \;\Rightarrow \mathbf{x}_{\hat{t}} = \mathbf{x}(\mathbf{X}) \tag{2.2}$$

The above expression concerns every material point from the reference state $\mathfrak{B}_0$ at time t_0 to a single point in the current (deformed) state $\mathfrak{B}_t$ at time $\hat{t}$. Now consider two neighboring material points $\boldsymbol{A}$ and $\boldsymbol{B}$ in $\mathfrak{B}_0$, with position vectors $\mathbf{X}$ and $\mathbf{X} + d\mathbf{X}$ respectively. The

corresponding position vectors on the deformed configuration at time $\hat{t}$ are $\mathbf{x}$ and $\mathbf{x} + d\mathbf{x}$ respectively. If we are able to determine the quantity $d\mathbf{x}$ in the deformed configuration, we will get an immediate sense of deformation infinitesimally close to the point $\mathbf{X}$. Thus, by differentiating (2.2) with respect to $\mathbf{X}$ we obtain:

$$\mathbf{x} = \mathbf{x}(\mathbf{X}) \;\Rightarrow d\mathbf{x} = \frac{\partial \mathbf{x}(\mathbf{X})}{\partial \mathbf{X}} \cdot d\mathbf{X}$$

The quantity $\partial \mathbf{x}(\mathbf{X})/\partial \mathbf{X}$ defines a second order tensor:

$$\boxed{\mathbf{F} = \mathbf{x}\nabla\mathbf{x} = \frac{\partial \mathbf{x}(\mathbf{X})}{\partial \mathbf{X}} \text{ or: } F_{ij} = \frac{\partial x_i}{\partial X_j}} \tag{2.3}$$

which is called the **Deformation Gradient**.

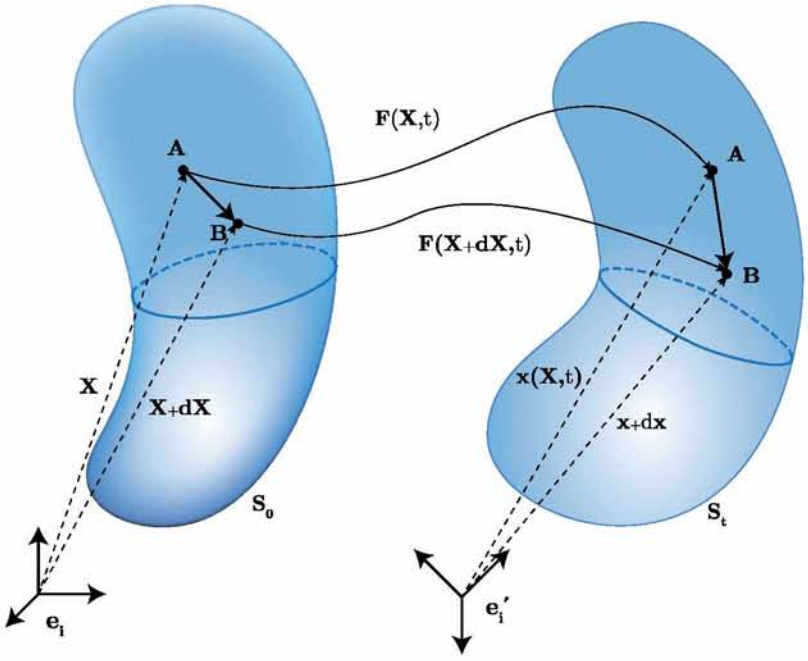


Fig. 2.2: Mapping through deformation gradient from the reference state to deformed state in the continuum body

The equation (2.3) can be written also in the form:

$$\boxed{d\mathbf{x}(\mathbf{X}, t) = \mathbf{F}(\mathbf{X}, t) \cdot d\mathbf{X} \qquad \text{(t=constant)}} \tag{2.4}$$

and it is clear that the deformation gradient is from definition the Jacobian matrix from $\mathfrak{B}_0$ to $\mathfrak{B}_t$ and therefore assigns 'quantities' from the reference to the current state.

The **Deformation Gradient** is one of the most important quantities defined in Continuum mechanics as it maps infinitesimal material fibers from the reference to the current (deformed) configuration which means that $\mathbf{F}$ contains all deformation information infinitesimally close to the point that fiber originates. Thus, it is reasonable to assume that changes in shape, size and orientation have to be expressed in terms of $\mathbf{F}$. Using tensor analysis someone will conclude on the above assumption that changes in length and relative orientation as well as surface and volume changes within the continuum body all given by expressions that involve the deformation gradient $\mathbf{F}$. Next, we briefly represent those geometric changes without prove them as it is beyond the scope of this discussion.

2.1.i Length changes

We consider two material points A and B in the reference and deformed state. The material points A and B are connected with the vector $d\mathbf{X}$ on the reference configuration and with vector $d\mathbf{x}$ on the deformed configuration. The relationship between vector $d\mathbf{X}$ and vector $d\mathbf{x}$ is given from the equation $d\mathbf{x} = \mathbf{F} \cdot d\mathbf{X}$. We introduce the infinitesimal lengths ds_0 and ds which can be expressed in terms of $d\mathbf{X}$ and $d\mathbf{x}$ respectively as:

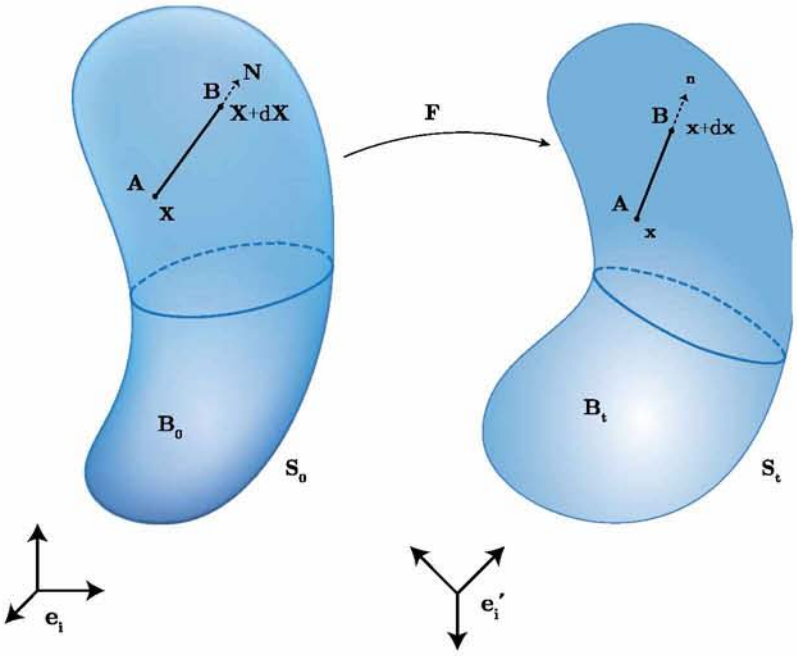


Fig. 2.3: Stretch of an infinitesimal material fiber

It can be proven that the stretch ratio λ of the material fiber defined as $\lambda = ds/ds_0$ is given by:

$$\lambda = \sqrt{\mathbf{N} \cdot \mathbf{F}^T \cdot \mathbf{F} \cdot \mathbf{N}} \quad \text{where:} \quad \mathbf{N} = \frac{d\mathbf{X}}{ds_0} \tag{2.5}$$

The tensor $\mathbf{F}^T \cdot \mathbf{F}$ is used widely in the analysis of continuum mechanics and it is known as

Right Cauchy-Green tensor which is given by:

$$\boxed{C = \mathbf{F}^T \cdot \mathbf{F}} \tag{2.6}$$

We also define the **Left Cauchy-Green** tensor:

$$\boxed{B = \mathbf{F} \cdot \mathbf{F}^T} \tag{2.7}$$

2.1.ii Angle changes

We consider two infinitesimal material fibers AB$(d\mathbf{X}_1, \mathbf{N}_1)$ and AC$(d\mathbf{X}_2, \mathbf{N}_2)$ on a continuum body, as shown in figure below 2.4. The angle in the deformed state is essentially the angle between the unit vectors $\mathbf{n}_1, \mathbf{n}_2$. Thus, the cosine is given by:

$$\cos\theta = \cos(\hat{\mathbf{n}}_1, \hat{\mathbf{n}}_2) = \frac{\mathbf{N}_1 \cdot \mathbf{N}_2}{\lambda_1 \lambda_2} \tag{2.8}$$

where λ_1 and λ_2 are the stretch ratios of material fibers AB and AC respectively.

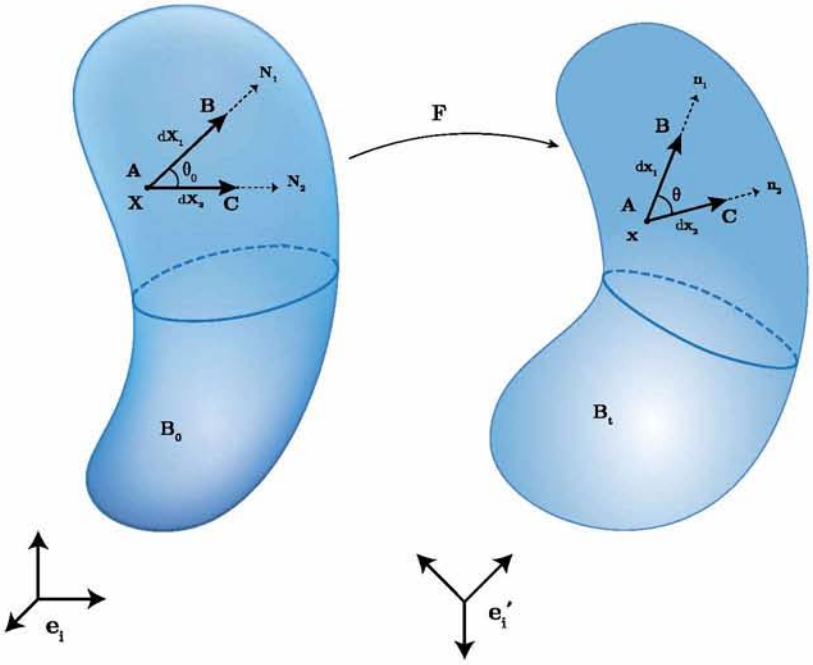


Fig. 2.4: Change of relative orientation between two infinitesimal material fibers

2.1.iii Surface changes

We consider an infinitesimal surface consisting of two infinitesimal material fibers $\vec{AB}$ and $\vec{AC}$ as shown in the figure. The normal to the infinitesimal surface unit vector $\hat{\mathbf{N}}$ defines

the orientation of the infinitesimal surface dS_0, which is forming from the material fibers $\vec{AB}$ and $\vec{AC}$ in the reference configuration and the vector $\hat{\mathbf{n}}$ defines the orientation of the infinitesimal surface dS, which is forming from the material fibers $\vec{AB}$ and $\vec{AC}$ in the deformed configuration. According to Nanson's formula the new infinitesimal surface dS and the new normal vector $\hat{\mathbf{n}}$ are given by:

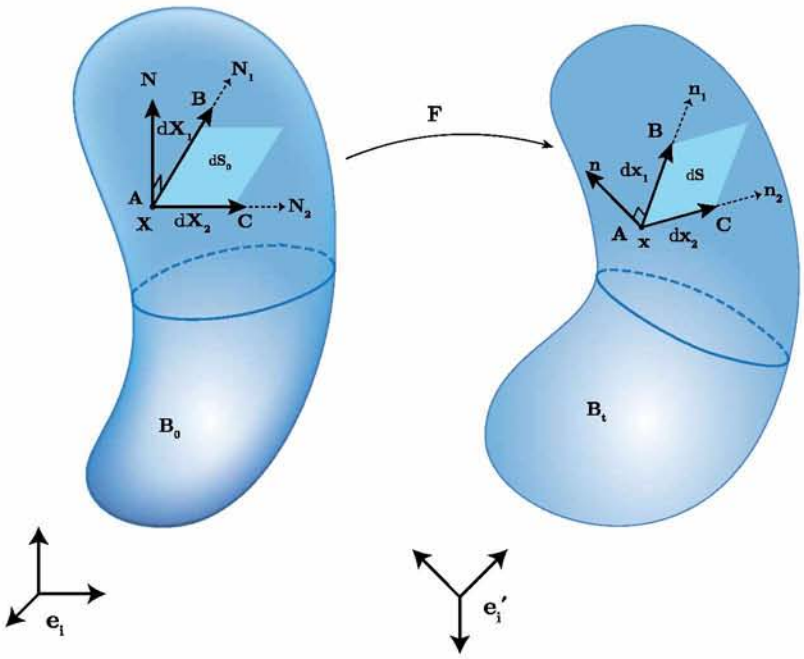


Fig. 2.5: Infinitesimal surface before and after deformation

$$\frac{dS}{dS_0} = (det\mathbf{F})\sqrt{\hat{\mathbf{N}} \cdot C^{-1} \cdot \hat{\mathbf{N}}} \tag{2.9}$$

$$\hat{\mathbf{n}} = \frac{1}{\sqrt{\hat{\mathbf{N}} \cdot C^{-1} \cdot \hat{\mathbf{N}}}} \cdot (\hat{\mathbf{N}} \cdot \mathbf{F}^{-1}) \tag{2.10}$$

2.1.iv Volume changes

We consider three infinitesimal material fibers $d\mathbf{X}^1$, $d\mathbf{X}^2$ and $d\mathbf{X}^3$. Changes in length of the three infinitesimal material fibers cause volume changes. Thus, the infinitesimal volume dV_0 in the reference state transformed into dV in the deform state as shown in the figure below.

It can be shown [23] that the infinitesimal volume dV_0 on the current configuration transforms into dV in the deformed configuration according to:

$$dV = J\ dV_0 = J\ det\mathbf{F} \tag{2.11}$$

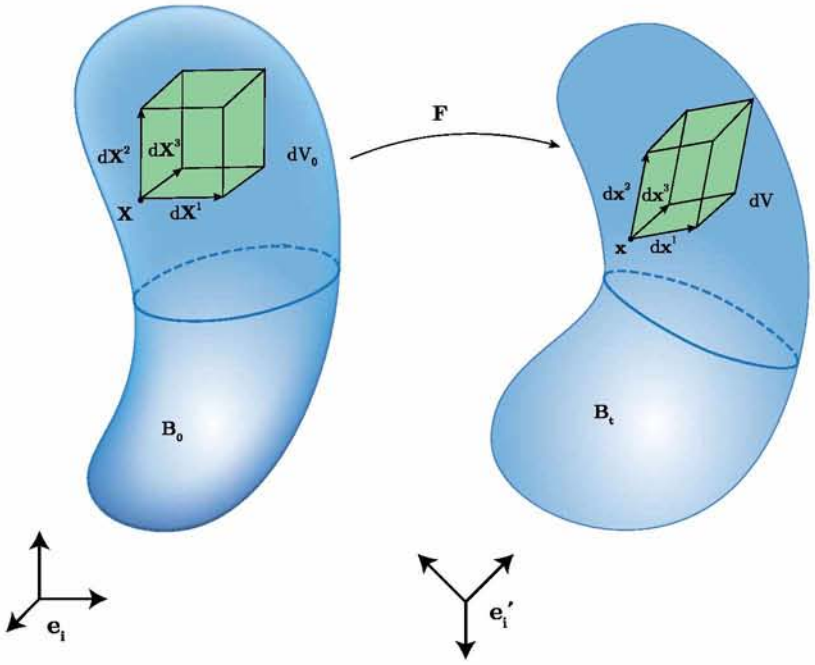


Fig. 2.6: Infinitesimal volume change

2.2 The Polar Decomposition Theorem

It can be shown [8] that an invertible second-order tensor $\boldsymbol{A}$ can be uniquely expressed as the product of a symmetric and an orthogonal tensor as:

$$\mathbf{A} = \mathbf{Q} \cdot \mathbf{U} = \mathbf{V} \cdot \mathbf{Q} \tag{2.12}$$

Now, in the context of continuum mechanics, $\mathbf{F}$ is a second-order invertible tensor and thus it can be decomposed as such:

$$\mathbf{F} = \mathbf{R} \cdot \mathbf{U} = \mathbf{V} \cdot \mathbf{R} \tag{2.13}$$

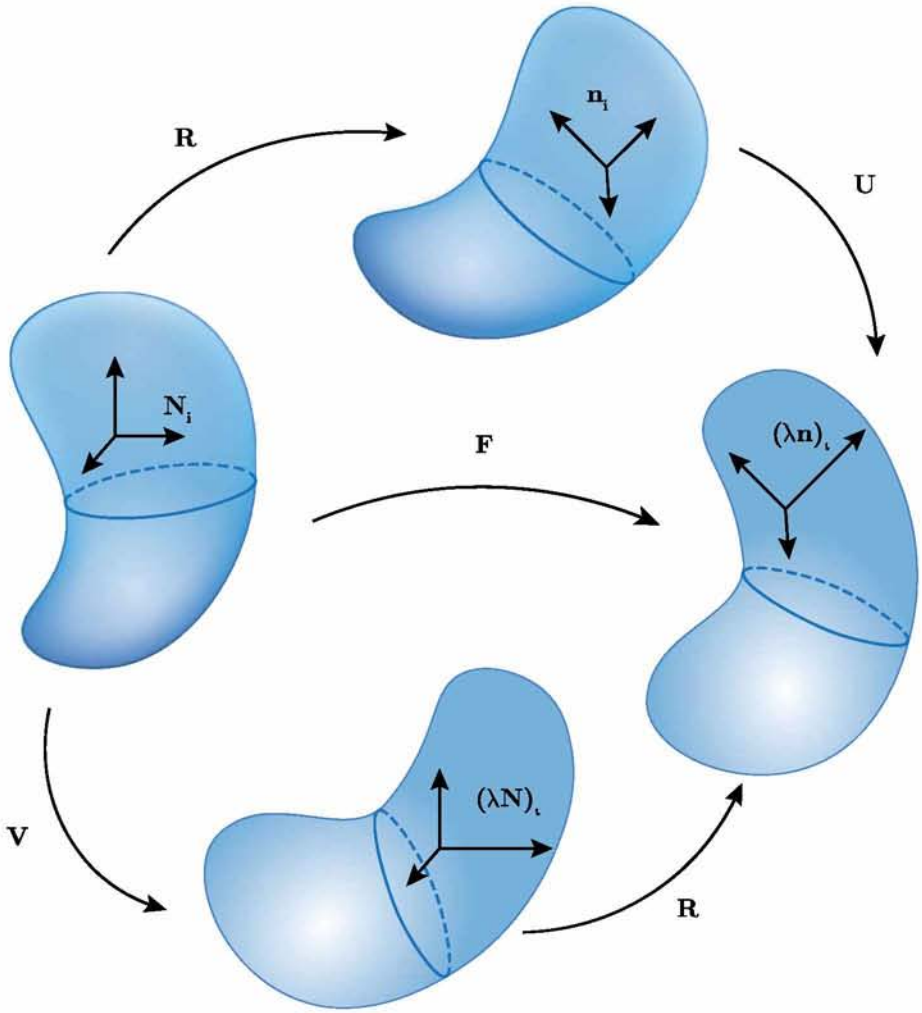


Fig. 2.7: The Polar Decomposition of **F**

where tensor **R** is an orthogonal tensor representing rigid rotations associated with the motion of **F**, whereas tensors **U** and **V** are symmetric and positive define and they express 'pure' deformation.

It can be shown [8] that tensors **R** , **U** , **V** are unique and expression (2.13) defines the **Polar Decomposition** of **F**. Tensor **U** is also commonly referred to as **left** stretch tensor and **V** as **right** stretch tensor. The fact that tensors **U** and **V** are symmetric and positive-define suggests that their eigenvalues are real and positive while also their eigenvectors define an orthonormal base. Let us denote as $\mathbf{N}_i$ the eigenvectors of **U** defining a **Lagrangian** triad

and as $\mathbf{n}_i$ the corresponding **Eulerian** base defined by the eigenvectors of $\mathbf{V}$. The left and right stretch tensors can now be expressed in terms of their principal directions as:

$$\mathbf{U} = \sum_{i=1}^{3} \lambda_i \, \mathbf{N}_i \, \mathbf{N}_i \tag{2.14a}$$

$$\mathbf{V} = \sum_{i=1}^{3} \lambda_i \, \mathbf{n}_i \, \mathbf{n}_i \tag{2.14b}$$

where λ_i are the eigenvalues of $\mathbf{U}$ and $\mathbf{V}$. Tensors $\mathbf{U}$ and $\mathbf{V}$ can also be expressed in terms of the right and left Cauchy–Green tensors that we defined in equations (2.6) and (2.7) as:

$$\mathbf{U} = \sqrt{\mathbf{C}} \quad , \quad \mathbf{V} = \sqrt{\mathbf{B}}$$

It is important to note however that the square root of a second order tensor can only be evaluated as the square root of the tensor's eigenvalues once the tensor is expressed in terms of a coordinate system whose axes are aligned with the tensor's principal directions. Therefore, the process of determining the square root of an arbitrary invertible matrix $\mathbf{A}$, always involves solving an eigenvalue problem. The expression of $\mathbf{U}$ and $\mathbf{V}$ as the square root of $\mathbf{C}$ and $\mathbf{B}$ respectively, suggests that the left and right Cauchy–Green tensors can also be expressed with respect to their principal directions as:

$$\mathbf{C} = \sum_{i=1}^{3} \lambda_i^2 \, \mathbf{N}_i \, \mathbf{N}_i$$

$$\mathbf{B} = \sum_{i=1}^{3} \lambda_i^2 \, \mathbf{n}_i \, \mathbf{n}_i$$

Finally, the orthogonal tensor $\mathbf{R}$ can be expressed in terms of the Eulerian $\mathbf{n}_i$ and Lagrangian $\mathbf{N}_i$ directors as:

$$\mathbf{R} = \mathbf{n}\mathbf{N} \tag{2.15}$$

2.3 Strain Measures

We have already discussed about deformation gradient $\mathbf{F}$ which contains all 'information' concerning deformation for every material point within the continuum body. Moreover, we have discussed about the polar decomposition of $\mathbf{F}$ which suggests that we can eliminate rigid rotation from $\mathbf{F}$ and express pure deformation in terms of the stretch tensors $\mathbf{U}$ and $\mathbf{V}$. Thus, it is reasonable to say that:

- Strain tensors must be coaxial [1] with either $\mathbf{U}$ or $\mathbf{V}$.

[1] Two arbitrary tensors $\mathbf{A}$ and $\mathbf{B}$ are coaxial if they can be diagonalized in the same coordinate system.

The above statement leads to the definition of two general strain measures:

$$\mathbf{E}^{(m)} = f^{(m)}(\lambda_i)\,\mathbf{N}_i\,\mathbf{N}_i \tag{2.16a}$$

$$\mathbf{e}^{(m)} = f^{(m)}(\lambda_i)\,\mathbf{n}_i\,\mathbf{n}_i \tag{2.16b}$$

All strain measures derived from (2.16a) are referred to as Lagrangian strains, while strain measures derived from (2.16b) are Eulerian[2] strains. Functions $f(\lambda_i)$ have to be chosen so that for small[3] strains, the strain measures being defined are consistent with the infinitesimal strain theory. Recalling that the stretch ratio λ is defined as the ratio of the current to the original length of a material fiber, the infinitesimal strain theory suggests:

$$\varepsilon = \frac{l - l_0}{l_0} = \frac{l}{l_0} - 1 = \lambda - 1 \tag{2.17}$$

Now, in the simplest possible case of uniaxial straining, we can expand $f^{(m)}(\lambda)$ around $\lambda = 1$ (small strains) to derive:

$$E(\lambda)|_{\lambda\to 1} = f^{(m)}(1) + \left.\frac{df^{(m)}(\lambda)}{d\lambda}\right|_{\lambda=1}(\lambda - 1) + O((\lambda-1)^2) \tag{2.18}$$

To ensure the aforementioned consistency, expression (2.18) must reduce to $\lambda - 1$. This demand leads to the following restrictions for the functions $f^{(m)}(\lambda_i)$:

$$f^{(m)}(1) = 0 \quad , \quad \left.\frac{df^{(m)}(\lambda)}{d\lambda}\right|_{\lambda=1} = 1$$

while simultaneously $f'(\lambda) > 0 \;\; \forall \;\; \lambda > 0$. A family of strains that satisfies all of the above constraints is defined as follows:

$$f^{(m)} = \begin{cases} \frac{1}{m}(\lambda^m - 1) & , \quad m \neq 0 \\ \ln\lambda & , \quad m = 0 \end{cases} \qquad m \in \mathbb{Z} \tag{2.19}$$

The most commonly encountered strain tensors derived from the above strain family are summarized below:

[2] Lagrangian Strains are coaxial with $\mathbf{U}$, while Eulerian strains are coaxial with $\mathbf{V}$

[3] Small strains correspond to $\lambda \cong 1$ or equivalently to $\lambda - 1 \cong 0$

Table 2.1: Strain measures commonly encountered in continuum mechanics applications

	Strain	Definition
$m = 0$	Logarithmic/Hencky	$\mathbf{E}^{(0)} = \ln \mathbf{U}$
		$\mathbf{e}^{(0)} = \ln \mathbf{V}$
$m = -2$	Almansi	$\mathbf{e}^{A} = \mathbf{e}^{(-2)} = \frac{1}{2}(\boldsymbol{\delta} - \mathbf{B}^{-1})$
$m = 2$	Green	$\mathbf{E}^{G} = \mathbf{E}^{(2)} = \frac{1}{2}(\mathbf{C} - \boldsymbol{\delta})$
$m = 1$	Biot	$\mathbf{E}^{B} = \mathbf{E}^{(1)} = \mathbf{U} - \boldsymbol{\delta}$

2.4 Rate of deformation

When studying the kinematics of deformation within a continuum body, apart from the deformation $\mathbf{F}$ itself we have to determine the rate of deformation within that body(we do the same when we study the dynamics of a moving body as we need to determine the velocity $\boldsymbol{v}(t)$ of the moving body apart from its position $\mathbf{x}(t)$). Thus we start our analysis by defining the velocity field that corresponds to the motion $\mathbf{x}(\mathbf{X}, t)$:

$$\boldsymbol{v}(\mathbf{x}(\mathbf{X}), t) = \frac{\partial \mathbf{x}(\mathbf{X}, t)}{\partial t} \tag{2.20}$$

In order to define the velocity field $d\boldsymbol{v}$ in an infinitesimal 'neighbourhood' $d\mathbf{x}$ around the point $\mathbf{x}$ we will consider time t as constant. Hence:

$$d\boldsymbol{v} = \boldsymbol{v}\left(\mathbf{x}(\mathbf{X} + d\mathbf{X}), t\right) - \boldsymbol{v}(\mathbf{x}(\mathbf{X}), t) = \frac{\partial \boldsymbol{v}(\mathbf{x}, t)}{\partial \mathbf{x}} \cdot d\mathbf{x} \tag{2.21}$$

The term $\partial \boldsymbol{v}(\mathbf{x}, t)/\partial \mathbf{x}$ on the above equation is the **Velocity Gradient** tensor and it is denoted by:

$$\boxed{\mathbf{L} = \boldsymbol{v} \boldsymbol{\nabla}_{\mathbf{x}} = \frac{\partial \boldsymbol{v}(\mathbf{x}, t)}{\partial \mathbf{x}}} \tag{2.22}$$

The definition of the Gradient Velocity tensor $\mathbf{L}$ help us to rewrite the expression for $d\boldsymbol{v}$ in the equation (2.21) as:

$$d\boldsymbol{v} = \mathbf{L} \cdot d\mathbf{x}$$

In the previous section 2.1 in equation (2.3) we analyzed the deformation gradient $\mathbf{F}$ which includes the appropriate information about the deformation of a continuum body. We have

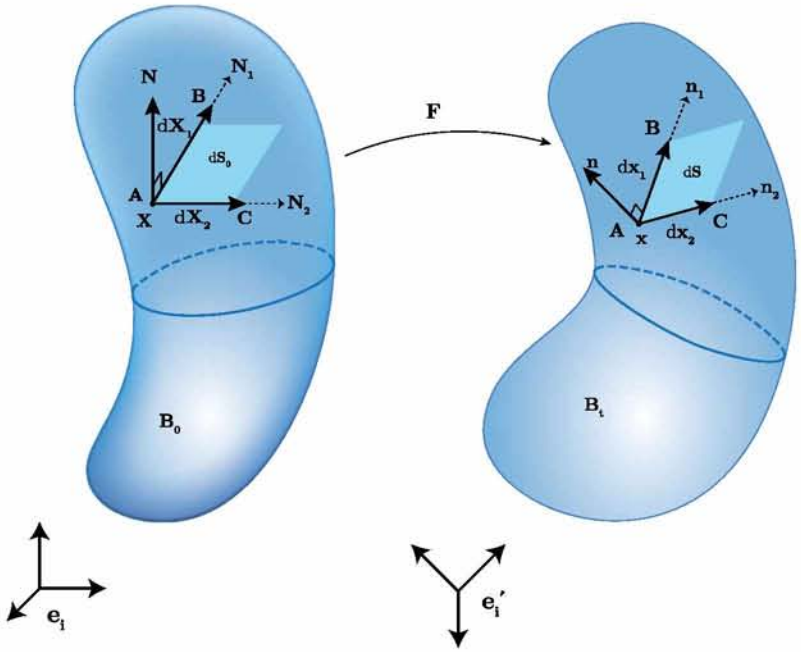


Fig. 2.8: Rate of deformation

already shown that $\mathbf{F}$ relates the vector $\mathbf{X}$ on the reference configuration with the vector $\mathbf{x}$ on the deformed configuration through equation $d\mathbf{x} = \mathbf{F} \cdot d\mathbf{X}$. Taking into account that $d\boldsymbol{v} = d\dot{\mathbf{x}}$ we can write:

$$d\mathbf{x} = \mathbf{F} \cdot d\mathbf{X} \;\Rightarrow\; \frac{\partial}{\partial t}(d\mathbf{x}) = \dot{\mathbf{F}} \cdot d\mathbf{X} \;\Rightarrow\; d\boldsymbol{v} = \dot{\mathbf{F}} \cdot \mathbf{F}^{-1} \cdot d\mathbf{x} \tag{2.23}$$

where:

$$\dot{\mathbf{F}} \cdot \mathbf{F}^{-1} = \frac{\partial}{\partial t}\left[\frac{\partial \mathbf{x}(\mathbf{X},t)}{\partial \mathbf{X}}\right] \cdot \frac{\partial \mathbf{X}}{\partial \mathbf{x}} = \frac{\partial}{\partial \mathbf{X}}\left[\frac{\partial \mathbf{x}(\mathbf{X},t)}{\partial t}\right] \cdot \frac{\partial \mathbf{X}}{\partial \mathbf{x}} = \frac{\partial \boldsymbol{v}(\mathbf{X},t)}{\partial \mathbf{x}} \tag{2.24}$$

Now taking into consideration equations (2.22) and (2.24) we can alternatively express the velocity gradient tensor in terms of the deformation gradient:

$$\boxed{\mathbf{L} = \dot{\mathbf{F}} \cdot \mathbf{F}^{-1}} \tag{2.25}$$

We know that any 2^{nd} order tensor can be expressed as the sum of a symmetric and an antisymmetric tensor. Thus, in the case of tensor $\mathbf{F}$ we can write:

$$\boxed{\mathbf{L} = \mathbf{D} + \mathbf{W}} \tag{2.26}$$

where $\mathbf{D}$ is the **Deformation Rate** and is defined as the symmetric part of $\mathbf{L}$ and $\mathbf{W}$ is the **Spin Tensor** and is defined as the antisymmetric part of $\mathbf{L}$. Therefore, we can write:

$$\mathbf{D} = \frac{1}{2}(\mathbf{L} + \mathbf{L}^T) \;\Rightarrow\; \mathbf{D} = \frac{1}{2}\left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right)\mathbf{e}_i\mathbf{e}_j \tag{2.27}$$

$$\mathbf{W} = \frac{1}{2}(\mathbf{L} - \mathbf{L}^T) \;\Rightarrow\; \mathbf{W} = \frac{1}{2}\left(\frac{\partial v_i}{\partial x_j} - \frac{\partial v_j}{\partial x_i}\right)\mathbf{e}_i\mathbf{e}_j \tag{2.28}$$

At this point we will describe briefly the physical interpretation of tensors $\mathbf{D}$ and $\mathbf{W}$.

2.4.i Physical interpretation of tensor D

In order to give the physical meaning of the components of the tensor $\mathbf{D}$ we need to consider an arbitrary infinitesimal material fiber $d\mathbf{X} = dS_0\mathbf{N}$ in the reference configuration which starts at point A and ends at point B. This material fiber transformed to $d\mathbf{x} = dS\mathbf{n}$ in the deformed configuration as shown in the figure below.

We are interested in determining the rate of change of the infinitesimal length ds. For this reason we differentiate ds with respect to time t:

$$ds = \sqrt{d\mathbf{x}\cdot d\mathbf{x}} \;\Rightarrow\; ds^2 = d\mathbf{x}\cdot d\mathbf{x} \;\Rightarrow\; \frac{\partial}{\partial t}(ds^2) = \frac{\partial}{\partial t}(d\mathbf{x}\cdot d\mathbf{x}) \;\Rightarrow$$

$$2ds\frac{\partial}{\partial t}(ds) = \underbrace{\frac{\partial}{\partial t}(d\mathbf{x})}_{d\mathbf{x}\cdot\mathbf{L}^T}\cdot d\mathbf{x} + d\mathbf{x}\cdot\underbrace{\frac{\partial}{\partial t}(d\mathbf{x})}_{\mathbf{L}\cdot d\mathbf{x}} = d\mathbf{x}\cdot\mathbf{L}^T\cdot d\mathbf{x} + d\mathbf{x}\cdot\mathbf{L}\cdot d\mathbf{x} \Rightarrow$$

$$d\mathbf{x}\cdot\underbrace{(\mathbf{L}^T + \mathbf{L})}_{2\mathbf{D}}\cdot d\mathbf{x} = 2d\mathbf{x}\cdot\mathbf{D}\cdot d\mathbf{x} \;\Rightarrow$$

$$\boxed{\frac{1}{ds}\frac{\partial}{\partial t}(ds) = \mathbf{n}\cdot\mathbf{D}\cdot\mathbf{n}} \tag{2.29}$$

From equation (2.29) we understand that the normal components of $\mathbf{D}$ (D_{nn}), are the rate of extension per unit length of a material fiber which, in the current configuration, is momentarily aligned with the direction of $\mathbf{n}$.

$$D_{nn} = \frac{1}{ds}\frac{\partial}{\partial t}(ds) = \frac{\partial}{\partial t}\left(\ln\frac{ds}{ds_0}\right) = \frac{\partial}{\partial t}(\ln\lambda) = \frac{\dot{\lambda}}{\lambda}$$

Now, in order to give the physical interpretation of the shear components of $\mathbf{D}$, we have to consider a material fiber $d\mathbf{x} = ds\mathbf{m}$ and define the rate of change of the unit vector $\mathbf{m}$ which is attached to the material fiber.

$$\mathbf{m} = \frac{d\mathbf{x}}{ds} \;\Rightarrow\; \dot{\mathbf{m}} = \frac{\partial}{\partial t}\left(\frac{d\mathbf{x}}{ds}\right) = \frac{1}{ds}\frac{\partial}{\partial t}(d\mathbf{x}) - \frac{d\mathbf{x}}{(ds)^2}\frac{\partial}{\partial t}(ds) \;\Rightarrow$$

$$\mathbf{L}\cdot\mathbf{m}-(\mathbf{m}\cdot\mathbf{D}\cdot\mathbf{m})\mathbf{m}=(\mathbf{W}+\mathbf{D})\mathbf{m}-(\mathbf{m}\cdot\mathbf{D}\cdot\mathbf{m})\mathbf{m}\ \Rightarrow$$

$$\dot{\mathbf{m}}=\mathbf{W}\cdot\mathbf{m}+\mathbf{D}\cdot\mathbf{m}-(\mathbf{m}\cdot\mathbf{D}\cdot\mathbf{m})\mathbf{m}=\mathbf{W}\cdot\mathbf{m}+\mathbf{D}\cdot\mathbf{m}\underbrace{\mathbf{m}\cdot\mathbf{m}}_{1}-\mathbf{m}(\mathbf{m}\cdot\mathbf{D}\cdot\mathbf{m})\Rightarrow$$

$$\boxed{\dot{\mathbf{m}}=\mathbf{W}^m\cdot\mathbf{m}=-\mathbf{m}\cdot\mathbf{W}^m}\ \text{ where: }\ \boxed{\mathbf{W}^m=\mathbf{W}+\mathbf{D}\cdot\mathbf{mm}-\mathbf{mm}\cdot\mathbf{D}} \tag{2.30}$$

The term $\mathbf{W}^m$ above is the spin of a unit vector $\mathbf{m}$ which is attached to a material fiber.

In order to complete our discussion about the physical interpretation of the shear components of $\mathbf{D}$, we consider two infinitesimal material fibers $d\mathbf{x}_1$ and $d\mathbf{x}_2$ at the current state as they are illustrated in figure 2.4. We can recall equation (2.8) in which the cosine of angle θ between $d\mathbf{x}_1$ and $d\mathbf{x}_2$ is determined. Thus we can write:

$$\mathbf{m}\cdot\mathbf{n}=\cos\theta\ \Rightarrow\ \dot{\theta}=-\frac{1}{\sin\theta}\frac{\partial}{\partial t}(\mathbf{m}\cdot\mathbf{n}) \tag{2.31}$$

We need to evaluate the derivative $\partial(\mathbf{m}\cdot\mathbf{n})/\partial t$ in terms so that we can determine the rate of change of the relative orientation between those two fibers.

$$\begin{aligned}\frac{\partial}{\partial t}(\mathbf{m}\cdot\mathbf{n})&=\underbrace{\dot{\mathbf{m}}}_{-\mathbf{m}\cdot\mathbf{W}^m}\cdot\mathbf{n}+\mathbf{m}\cdot\underbrace{\dot{\mathbf{n}}}_{\mathbf{W}^n\cdot\mathbf{n}}=-\mathbf{m}\cdot\mathbf{W}^m\cdot\mathbf{n}+\mathbf{m}\cdot\mathbf{W}^n\cdot\mathbf{n}=\mathbf{m}\cdot(\mathbf{W}^n-\mathbf{W}^m)\cdot\mathbf{n}\\&=\mathbf{m}\cdot(\mathbf{D}\cdot\mathbf{nn}-\mathbf{nn}\cdot\mathbf{D}-\mathbf{D}\cdot\mathbf{mm}+\mathbf{mm}\cdot\mathbf{D})\cdot\mathbf{n}\\&=\mathbf{m}\cdot\mathbf{D}\cdot\mathbf{n}-\cos\theta D_{nn}-D_{mm}\cos\theta+\mathbf{m}\cdot\mathbf{D}\cdot\mathbf{n}\\&=2\mathbf{m}\cdot\mathbf{D}\cdot\mathbf{n}-(D_{mm}+D_{nn})\cos\theta\end{aligned} \tag{2.32}$$

Using (2.31) into (2.32) we derive:

$$\boxed{\dot{\theta}=\frac{1}{\sin\theta}\left[(D_{mm}+D_{nn})\cos\theta-2\mathbf{m}\cdot\mathbf{D}\cdot\mathbf{n}\right]} \tag{2.33}$$

Now we set on equation (2.33) $\mathbf{m}=\mathbf{e}_1$ and $\mathbf{n}=\mathbf{e}_2$ (which means that the vectors $\mathbf{m}$ and $\mathbf{n}$ are perpendicular to each other) to simplify the expression so that we can derive the physical interpretation of the shear components of tensor $\mathbf{D}$.

$$\mathbf{m}\perp\mathbf{n}\ (\theta=90^o)\ \Rightarrow\ \cos\theta=0\ \text{and}\ \sin\theta=1\ \overset{(2.33)}{\Rightarrow}\dot{\theta}=-2\mathbf{e}_1\cdot\mathbf{e}_2=-2\mathbf{D}_{12}\ \Rightarrow$$

$$\boxed{\mathbf{D}_{12}=-\frac{1}{2}\dot{\theta}\quad\text{or}\quad 2\mathbf{D}_{12}=-\dot{\theta}}$$

On the above equation, the term $2\mathbf{D}_{12}$ is the rate of decrease of the angle between a pair of material fibers which, in the current configuration, intersect at $\mathbf{x}$ and are momentarily

aligned with the directions of $\mathbf{e}_1$ and $\mathbf{e}_2$. This also implies that the rate of change of the relative orientation of the material fibers which, in the current configuration, are momentarily aligned with principal directions of $\mathbf{D}$, is zero.

It is worth mentioning that although the components of $\mathbf{D}$ can be though to express the rate of deformation infinitesimally close to the point of interest, there is no strain tensor $\mathbf{E}$ such that $\dot{\mathbf{E}} = \mathbf{D}$.

2.4.ii Physical interpretation of tensor W

Now we will try to give the physical interpretation of tensor $\mathbf{W}$ which is the antisymmetric part of the tensor $\mathbf{L}$ as mentioned in section 2.1. In order to do that we need to recall the rate of change of an arbitrary unit vector $\mathbf{m}$, so recall equation (2.30). If we consider that the unit vector $\mathbf{m}$ is along a material fiber that is momentarily aligned with one of the principal directions of $\mathbf{D}$, then $\mathbf{W}^m = \mathbf{W}$ and this result implies that the spin tensor $\mathbf{W}$ could be though as the spin of the material fibers that instantaneously coincide with the principal directions of $\mathbf{D}$.

Recalling that $\mathbf{W}$ is from definition an anti-symmetric tensor we can write the following:

$$\mathbf{W} = \omega_3(-\mathbf{e}_1\mathbf{e}_2 + \mathbf{e}_2\mathbf{e}_1) + \omega_1(-\mathbf{e}_2\mathbf{e}_3 + \mathbf{e}_3\mathbf{e}_2) + \omega_2(-\mathbf{e}_3\mathbf{e}_1 + \mathbf{e}_1\mathbf{e}_3) \tag{2.34}$$

where $\mathbf{e}_i$ are the unit vectors along the coordinate axes. Now we will express the arbitrary unit vector $\mathbf{m}$ in terms of the unit vector $\mathbf{e}_i$.

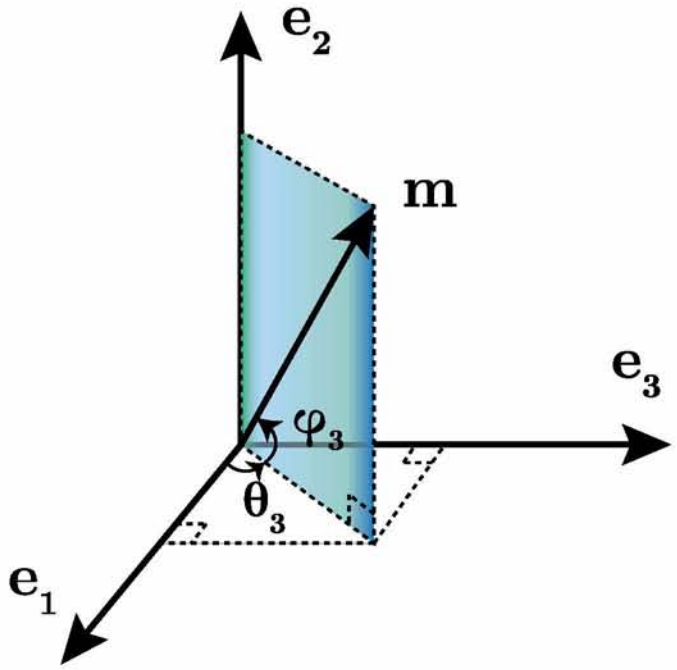


Fig. 2.9: Vector $\mathbf{m}$ *in the current configuration*

$$\mathbf{m} = \cos\theta_3 \cos\phi_3 \mathbf{e}_1 + \sin\theta_3 \cos\phi_3 \mathbf{e}_2 + \sin\phi_3 \mathbf{e}_3 \tag{2.35}$$

Now, if we try to calculate the rate of change of angle θ_3 we will conclude on the equation below:

$$\begin{aligned}\dot{\theta_3} = \omega_3 + \tan\phi_3(-\omega_2 \sin\theta_3 + \omega_1 \cos\theta_3) + \frac{1}{2}(D_{22} - D_{11}) \sin 2\theta_3 + D_{12} \cos 2\theta_3 \\ + \tan\phi_3(-D13 \sin\theta_3 + D_{23} \cos\theta_3)\end{aligned} \tag{2.36}$$

The local mean rate of rotation about the x_3-axis is defined as:

$$< \dot{\theta}_3 >= \frac{1}{2\pi^2} \int_{-\pi/2}^{\pi/2} \int_{0}^{2\pi} \dot{\theta}_3 \, d\theta_3 \, d\phi \quad \overset{(2.36)}{\Rightarrow} \quad \boxed{< \dot{\theta}_3 >= \omega_3} \tag{2.37}$$

Now, calculating the local mean rate of rotation about the x_1 and x_2 axes in combination with the rate of change of θ_1 and θ_2 respectively, it can be shown that:

$$\boxed{< \dot{\theta}_1 >= \omega_1} \tag{2.38}$$

$$\boxed{< \dot{\theta}_2 >= \omega_2} \tag{2.39}$$

At this point, according to Aravas and Aifantis [7], equations (2.37),(2.38) and (2.39) demonstrate that the spin tensor $\mathbf{W}$ is the average spin of all directions around a material point.

2.5 Cauchy stress tensor

Let us consider a general deformable body at its current position. As shown in the figure below, the traction vector $\mathbf{t}$ corresponding to the normal $\mathbf{n}$ is defined by:

$$\mathbf{t}(\mathbf{n}) = \lim_{\Delta\alpha \to 0} \frac{\Delta \mathbf{p}}{\Delta \alpha} \tag{2.40}$$

According to Newton's 3^{rd} law the relationship between vectors $\mathbf{t}$ and $\mathbf{n}$ is expressed as:

$$\mathbf{t}(-\mathbf{n}) = -\mathbf{t}(\mathbf{n}) \tag{2.41}$$

In order to define the stress tensor we will express the traction vectors in component form with respect to the three Cartesian directions $\mathbf{e}_1, \mathbf{e}_2$ and $\mathbf{e}_3$:

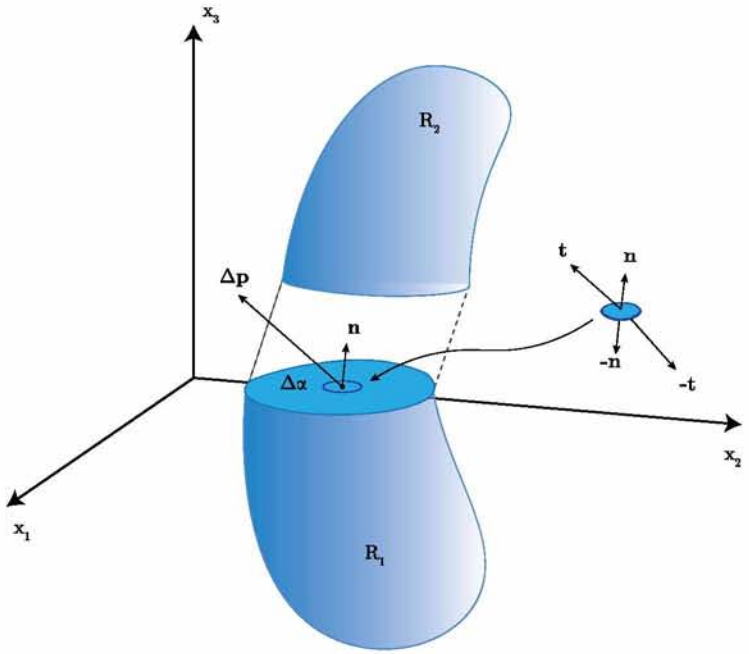


Fig. 2.10: Traction vector $\mathbf{t}$

$$\mathbf{t}(e_1) = \sigma_{11}\ \mathbf{e}_1 + \sigma_{12}\ \mathbf{e}_2 + \sigma_{13}\ \mathbf{e}_3$$

$$\mathbf{t}(e_2) = \sigma_{21}\ \mathbf{e}_1 + \sigma_{22}\ \mathbf{e}_2 + \sigma_{23}\ \mathbf{e}_3$$

$$\mathbf{t}(e_3) = \sigma_{31}\ \mathbf{e}_1 + \sigma_{32}\ \mathbf{e}_2 + \sigma_{33}\ \mathbf{e}_3$$

or in vector form

$$\mathbf{t}(e_i) = \sigma_{i1}\ \mathbf{e}_1 + \sigma_{i2}\ \mathbf{e}_2 + \sigma_{i3}\ \mathbf{e}_3 \tag{2.42}$$

In order to define the relationship between the traction vector $\mathbf{t}$ corresponding to a general direction $\mathbf{n}$ and the components σ_{ij}, we consider the Cauchy tetrahedron. Assuming that $\boldsymbol{f}$ is the force per unit volume acting on the body at point p, then the equilibrium of the tetrahedron is given as:

$$\mathbf{t}(\mathbf{n})d\alpha + \sum_{i=1}^{3} \mathbf{t}(-\mathbf{e}_i)d\alpha_i + \mathbf{dv} = 0 \quad \overset{:\mathbf{d}\alpha}{\rightarrow}$$

$$\mathbf{t}(\mathbf{n}) + \sum_{j=1}^{3} \mathbf{t}(-\mathbf{e}_j)\frac{d\alpha_j}{d\alpha} - \boldsymbol{f}\frac{dv}{d\alpha} = 0$$

Recalling Newton's 3^{rd} law and noting that $dV/d\alpha \rightarrow 0$ we can write the following:

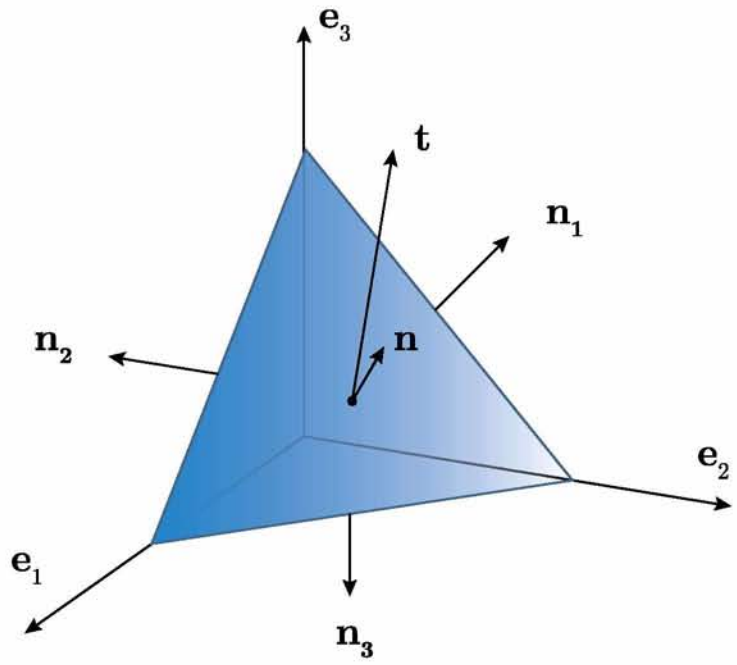


Fig. 2.11: The traction vector $\mathbf{t}$ *on the Cauchy tetrahedron*

$$\mathbf{t}(\mathbf{n}) = \sum_{j=1}^{3} \mathbf{t}(\mathbf{e}_j)(\mathbf{n}\cdot\mathbf{e}_j) = \sum_{i,j=1}^{3} \sigma_{ij}(\mathbf{e}_j\cdot\mathbf{n})\mathbf{e}_i$$

$$\mathbf{t}(\mathbf{n}) = \left[\sum_{i,j=1}^{3} \sigma_{ij}(\mathbf{e}_i\mathbf{e}_j)\right]\mathbf{n} \tag{2.43}$$

Equation (2.43) clearly identifies a tensor $\boldsymbol{\sigma}$ or the *Cauchy stress tensor* [4] , that relates the normal vector $\mathbf{n}$ to the traction vector $\mathbf{t}$ as:

$$\boxed{\mathbf{t}(\mathbf{n}) = \boldsymbol{\sigma}\mathbf{n}; \qquad \boldsymbol{\sigma} = \sum_{i,j=1}^{3} \sigma_{ij}\mathbf{e}_i\mathbf{e}_j} \tag{2.44}$$

2.5.i Stress Objectivity

In order to define the concept of Objectivity, we will consider the motion $\mathbf{x} = \mathbf{x}(\mathbf{X}, t)$ with a deformation gradient $\mathbf{F}(\mathbf{X}, t)$ and superimpose a rigid body motion denoted with $\mathbf{c(t)}$, so that

$$\tilde{\mathbf{x}}(\mathbf{X}, t) = \mathbf{Q}(t)\cdot\mathbf{x}(\mathbf{X}, t) + \boldsymbol{c}(t)$$

[4] A more detailed proof about the Cauchy stress tensor can be found on Aravas[6] and Bonet[11]

where $\mathbf{Q}(t)$ is an orthogonal tensor describing the superimposed rigid body rotation (Figure 2.12). Now, by differentiating the above equation we can obtain the deformation gradient corresponding to the new motion, $\bar{\mathbf{F}}$:

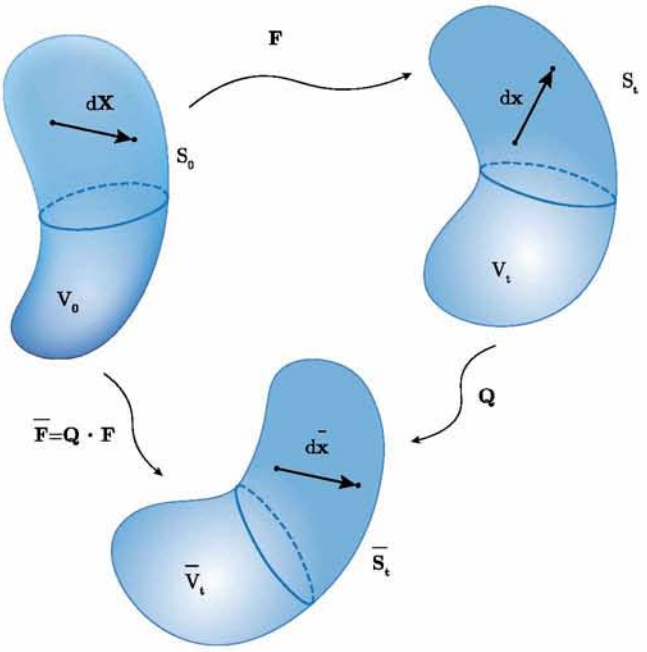


Fig. 2.12: Superimposed rigid body motion

$$\tilde{\mathbf{F}}(\mathbf{X},t) = \mathbf{Q}(t)\cdot\mathbf{F}(\mathbf{X},t) \tag{2.45}$$

Assuming now that $\mathbf{x}$ and $\bar{\mathbf{x}}$ are recorded by two different observers we can define objective tensors in the following sense:

1. Lagrangian tensors defined in $\mathfrak{B}_0$ are objective if only they remains unaffected by the observer's motion in the sense:

$$\tilde{\mathbf{a}}(\mathbf{X},t) = \mathbf{a}(\mathbf{X},t)$$
$$\tilde{\mathbf{A}}(\mathbf{X},t) = \mathbf{A}(\mathbf{X},t)$$

2. Eulerian tensors defined in $\mathfrak{B}_t$ are objective if only they transform according to:

$$\tilde{\mathbf{a}}(\bar{\mathbf{x}},t) = \mathbf{Q}(t)\cdot\mathbf{a}(\mathbf{X},t)$$
$$\tilde{\mathbf{A}}(\bar{\mathbf{x}},t) = \mathbf{Q}(t)\cdot\mathbf{A}(\mathbf{X},t)\cdot\mathbf{Q}^T(t)$$

3. Two-point second order tensors are objective if only they transform according to:

$$\tilde{\mathbf{A}} = \mathbf{Q}(t)\cdot\mathbf{A} \text{ or } \tilde{\mathbf{A}} = A\cdot\mathbf{Q}^T(t)$$

Using the definition above for an objective tensor it is easy to demonstrate that the Cauchy stress, $\boldsymbol{\sigma}$ is an objective tensor. For this purpose, consider the transformations of the normal and traction vectors implied by the superimposed rigid body motion $\mathbf{Q}$ as:

$$\tilde{\mathbf{t}}(\tilde{\mathbf{n}}) = \mathbf{Q} \cdot \mathbf{t}(\mathbf{n});$$
$$\tilde{\mathbf{n}} = \mathbf{Q} \cdot \mathbf{n}$$

Using equation (2.44) in conjunction with the above equation gives:

$$\tilde{\boldsymbol{\sigma}} = \mathbf{Q}\boldsymbol{\sigma}\mathbf{Q}^T \tag{2.46}$$

which conforms with the definition of objectivity and hence $\boldsymbol{\sigma}$ is an objective tensor.

2.6 Stress measures

We have already defined several strain measures section 2.3. Now we need to define the corresponding stress measures. The reason behind the definition of these strain and stress measures is that we hypothesized that the reference and the current states may, in general, be significant different.

We begin by defining the **Kirchhoff stress** in the current configuration:

$$\boldsymbol{\tau} = J\boldsymbol{\sigma} \tag{2.47}$$

The Kirchhoff stress is used widely in numerical algorithms in metal plasticity where there is no change in volume during plastic deformation (chapter 1, section 1.1.i). Kirchhoff stress, similarly to the Cauchy stress, is defined in the current state and is therefore a measure of force per unit deformed area. Since the undeformed geometry is known during deformation, we prefer to use unit vector $\mathbf{N}$ of a surface element in the reference state instead of $\mathbf{n}$, as $\mathbf{N}$ is easier to be determined [5]. The fact that we use vector $\mathbf{N}$ instead of $\mathbf{n}$ leads to the definition of the **Nominal stress** as:

$$\boldsymbol{T} = J\mathbf{F}^{-1} \cdot \boldsymbol{\sigma} \tag{2.48}$$

The Nominal stress is derived from Cauchy stress by expressing the unit vector $\mathbf{n}$ in terms of the unit vector $\mathbf{N}$ (Recall equation (2.10)). In this case the component T_{ij} declares the $j-th$ component of force per unit area in the reference configuration, on a surface element

[5] Unit vector $\mathbf{n}$ of a surface element in the current state is constantly changing with deformation, thus it is more difficult to be determined.

of the current configuration whose normal was in the i direction in the undeformed state. Moreover, from definition the Nominal stress is not symmetric since it has been expressed in terms of $\mathbf{F}$ which is not symmetric.

We proceed with the definition of two other stress measures, the **1st** and the **2nd Piola-kirchhoff**:

- The **1st Piola-Kirchhoff, P,** is the transpose of the Nominal stress and is defined by:

$$P = J\boldsymbol{\sigma}^T \cdot \mathbf{F}^{-T} = \boldsymbol{T}^T \tag{2.49}$$

- The **2nd Piola-Kirchhoff, S,** is defined in the reference configuration and expressed in terms of the Kirchhoff, $\boldsymbol{\tau}$, and Cauchy stress, $\boldsymbol{\sigma}$,as:

$$\mathbf{S} = \mathbf{F}^{-1} \cdot \boldsymbol{\tau} \cdot \mathbf{F}^{-T} = J\mathbf{F}^{-1} \cdot \boldsymbol{\sigma} \cdot \mathbf{F}^{-T} \tag{2.50}$$

Finally, we define the **Mandel stress** as:

$$\boldsymbol{\Sigma} = \boldsymbol{C}^e \cdot \boldsymbol{S}^e \tag{2.51}$$

where $\boldsymbol{C}^e$ is the Right Cauchy-Green tensor corresponding to $\mathbf{F}^e$ and $\boldsymbol{S}^e$ is the 2nd Piola-Kirchhoff elastic stress.

Table 2.2: Work Rate Conjugate Stress–Strain rate pairs

Stress	Strain Rate	Work Rate
$\boldsymbol{\sigma}$	$\mathbf{D}$	$\sigma_{ij}D_{ij}$
$\boldsymbol{\tau}$	$\mathbf{D}$	$\tau_{ij}D_{ij}$
$\mathbf{T}$	$\dot{\mathbf{F}}$	$T_{ij}\dot{F}_{ij}$
$\mathbf{P}$	$\dot{\mathbf{F}}^T$	$P_{ij}\dot{F}_{ji}$
$\mathbf{S}$	$\dot{\mathbf{E}}^G$	$S_{ij}\dot{E}^G_{ij}$
$\boldsymbol{\Sigma}$	$\dot{\mathbf{E}}^G$	$\Sigma_{ij}\dot{E}^G_{ij}$

Chapter 3

Constitutive Model

3.1 Elastic and plastic constitutive equations

In this chapter we will present governing equations for the elastic and plastic behaviour of isotropic and homogenous metallic materials undergoing finite deformations. We will also discuss the integration schemes used in order to integrate those equations numerically. We begin by writing the *Polar Decomposition Theorem* in terms of the elastic parts of the tensors involved, $\mathbf{R}$ and $\mathbf{U}$:

$$\mathbf{F}^e = \mathbf{R}^e \cdot \mathbf{U}^e \tag{3.1}$$

In the case of metals we assume small elastic strains as they are much smaller compared to the plastic one. Then, this assumption can be declared in the model through $\mathbf{U}^e$ by introducing a small quantity as shown in equations below:

$$\mathbf{U}^e = \boldsymbol{\delta} + \varepsilon \mathbf{A} \quad \text{where:} \quad |\varepsilon| \ll 1$$

whereas the quantity $\mathbf{A}$ is of order $\mathbf{A}^T = \mathbf{A} = O(1)$.

Now, we introduce the *velocity gradient* as $\mathbf{L}^e = \dot{\mathbf{F}}^e \cdot \mathbf{F}^{e-1}$. Let us determine the quantities involved in this last expression.

- Evaluation of $\dot{\mathbf{F}}^e$

$$\dot{\mathbf{F}}^e = \dot{\mathbf{R}}^e \cdot (\boldsymbol{\delta} + \varepsilon \mathbf{A}) + \mathbf{R}^e \cdot (\varepsilon \dot{\mathbf{A}}) = \dot{\mathbf{R}}^e + \varepsilon(\dot{\mathbf{R}}^e \cdot \mathbf{A} + \mathbf{R}^e \cdot \dot{\mathbf{A}})$$

- Evaluation of $\mathbf{F}^{e-1}$

$$\mathbf{F}^{e-1} = \mathbf{U}^{e-1} \cdot \mathbf{R}^{eT} = \left[\boldsymbol{\delta} - \varepsilon\mathbf{A} + O(\varepsilon^2)\right] \cdot \mathbf{R}^{eT} = \mathbf{R}^{eT} - \varepsilon\mathbf{A} \cdot \mathbf{R}^{eT} + O(\varepsilon^2)$$

Now, we substitute the above expressions into $\mathbf{L}^e$:

$$\mathbf{L}^e = \dot{\mathbf{F}}^e \cdot \mathbf{F}^{e-1} = \left[\dot{\mathbf{R}}^e + \varepsilon\left(\dot{\mathbf{R}}^e \cdot \mathbf{A} + \mathbf{R}^e \cdot \dot{\mathbf{A}}\right)\right] \cdot \left[\mathbf{R}^{eT} - \varepsilon\mathbf{A} \cdot \mathbf{R}^{eT} + O(\varepsilon^2)\right] \rightarrow$$

$$\mathbf{L}^e = \underbrace{\dot{\mathbf{R}}^e \cdot \mathbf{R}^{eT}}_{\boldsymbol{\omega}} + \varepsilon \underbrace{\mathbf{R}^e \cdot \dot{\mathbf{A}} \cdot \mathbf{R}^{eT}}_{symmetric} + O(\varepsilon^2\dot{\mathbf{R}}^e, \varepsilon^2\dot{\mathbf{A}}) \tag{3.2}$$

Moreover, we know from chapter 2 that $\mathbf{D}$ is the symmetric part of the $\mathbf{L}$ whereas $\mathbf{W}^*$ is the antisymmetric part of $\mathbf{L}$. Thus, it is easy to calculate $\mathbf{D}^e$ and $\mathbf{W}^*$.

- $\mathbf{D}^e = \mathbf{L}^e_{sym} = \varepsilon\mathbf{R}^e \cdot \dot{bA} \cdot \mathbf{R}^{eT} + O\left(\varepsilon^2\dot{\mathbf{R}}^e, \varepsilon^2\dot{\mathbf{A}}\right) = \underbrace{\mathbf{R}^e \cdot \dot{\mathbf{U}}^e \cdot \mathbf{R}^{eT}}_{O(\varepsilon)} + O\left(\varepsilon^2\dot{\mathbf{R}}^e, \varepsilon^2\dot{\mathbf{A}}\right) \rightarrow$

$$\boxed{\mathbf{D}^e \cong \underbrace{\mathbf{R}^e \cdot \dot{\mathbf{U}}^e \cdot \mathbf{R}^{eT}}_{O(\varepsilon\dot{\mathbf{A}})}}$$

- $\mathbf{W}^* = \mathbf{L}^e_{antisym} = \boldsymbol{\omega} + O\left(\varepsilon^2\dot{\mathbf{R}}^e, \varepsilon^2\dot{\mathbf{A}}\right) \rightarrow \quad \boxed{\mathbf{W}^* \cong \boldsymbol{\omega} = \dot{\mathbf{R}}^e \cdot \mathbf{R}^{eT}}$

Recall from chapter 2 the definition of two stress measures, the 2^{nd} **Piola-Kirchhoff**, (2.50), and **Mandel**,(2.51). We proceed by writing the elastic part of those stress measures:

- Let us begin with the evaluation of the $2^{nd} Piola - Kirchhoff$:

$$\mathbf{S}^e = J^e\mathbf{F}^{e-1} \cdot \boldsymbol{\sigma} \cdot \mathbf{F}^{e-T} = \mathbf{R}^{eT} \cdot \boldsymbol{\sigma} \cdot \mathbf{R}^e + O(\varepsilon\boldsymbol{\sigma}) \rightarrow \quad \mathbf{S}^e \cong \mathbf{R}^{eT} \cdot \boldsymbol{\sigma} \cdot \mathbf{R}^e \tag{3.3}$$

- We continue with the evaluation of the **Mandel stress**:

$$\boldsymbol{\Sigma}^e = J^e\mathbf{F}^{e-1} \cdot \boldsymbol{\sigma} \cdot \mathbf{F}^e = \mathbf{R}^{eT} \cdot \boldsymbol{\sigma} \cdot \mathbf{R}^e + O(\varepsilon\boldsymbol{\sigma}) \rightarrow \quad \boldsymbol{\Sigma}^e \cong \mathbf{R}^{eT} \cdot \boldsymbol{\sigma} \cdot \mathbf{R}^e \tag{3.4}$$

From equations (3.3) and (3.4) we can conclude that:

$$\boxed{\mathbf{S}^e \cong \boldsymbol{\Sigma}^e \cong \mathbf{R}^{eT} \cdot \boldsymbol{\sigma} \cdot \mathbf{R}^e = \text{symmetric}} \tag{3.5}$$

Now it is reasonable to take the rates of the aforementioned stress measures as they will be prove to be very useful in our analysis.

$$\dot{\mathbf{S}}^e = \dot{\boldsymbol{\Sigma}}^e = \dot{\mathbf{R}}^{eT} \cdot \boldsymbol{\sigma} \cdot \mathbf{R}^e + \mathbf{R}^{eT} \cdot \dot{\boldsymbol{\sigma}} \cdot \mathbf{R}^e + \mathbf{R}^{eT} \cdot \boldsymbol{\sigma} \cdot \dot{\mathbf{R}}^e + O(\varepsilon\dot{\boldsymbol{\sigma}}, \varepsilon\boldsymbol{\sigma} \cdot \dot{\mathbf{R}}) =$$

$$= \mathbf{R}^{eT}\left(\underbrace{\mathbf{R}^{e}\cdot\dot{\mathbf{R}}^{eT}}_{-\boldsymbol{\omega}}\cdot\boldsymbol{\sigma}+\dot{\boldsymbol{\sigma}}+\boldsymbol{\sigma}\cdot\underbrace{\dot{\mathbf{R}}^{e}\cdot\mathbf{R}^{eT}}_{\boldsymbol{\omega}}\right)\cdot\mathbf{R}^{e}+O(\varepsilon\dot{\boldsymbol{\sigma}},\varepsilon\boldsymbol{\sigma}\cdot\dot{\mathbf{R}})\Rightarrow$$

$$\boxed{\dot{\mathbf{S}}^{e}\cong\dot{\boldsymbol{\Sigma}}^{e}\cong\mathbf{R}^{eT}\cdot\overset{o}{\boldsymbol{\sigma}}\cdot\mathbf{R}^{e}} \tag{3.6}$$

where $\overset{o}{\boldsymbol{\sigma}}$ is a stress rate co-rotational with the substructure and is given by:

$$\overset{o}{\boldsymbol{\sigma}}=\dot{\boldsymbol{\sigma}}+\boldsymbol{\sigma}\cdot\boldsymbol{\omega}-\boldsymbol{\omega}\cdot\boldsymbol{\sigma} \tag{3.7}$$

The conjugate strain measure for the 2^{nd} **Piola-Kircchoff stress** and the **Mandel stress** is the *Green* strain tensor which is given by $\mathbf{E}^{G}=\frac{1}{2}(\mathbf{C}-\boldsymbol{\delta})$. Now we will proceed to the determination of the rate of Green strain.

$$\dot{\mathbf{E}}^{Ge}=\mathbf{F}^{eT}\cdot\mathbf{D}^{e}\cdot\mathbf{F}^{e}=\mathbf{R}^{eT}\cdot\mathbf{D}^{e}\cdot\mathbf{R}^{e}+O(\varepsilon\mathbf{D}^{e})\;\Rightarrow\;\boxed{\dot{\mathbf{E}}^{Ge}\cong\mathbf{R}^{eT}\cdot\mathbf{D}^{e}\cdot\mathbf{R}^{e}} \tag{3.8}$$

At this point, we have already evaluated all the appropriate quantities to write the equations that govern the elastic behavior.

Elastic equations:

$$\dot{\mathbf{S}}^{e}=\mathbf{L}^{e}:\dot{\mathbf{E}}^{Ge}\overset{(3.6)}{\underset{(3.8)}{\Rightarrow}}$$

$$\mathbf{R}^{eT}\cdot\overset{o}{\boldsymbol{\sigma}}\cdot\mathbf{R}^{e}\cong\mathbf{L}^{e}:(\mathbf{R}^{eT}\cdot\mathbf{D}^{e}\cdot\mathbf{R}^{e})\;\Rightarrow\;\overset{o}{\boldsymbol{\sigma}}\cong\mathbf{R}^{e}\cdot[\mathbf{L}^{e}:(\mathbf{R}^{e}T\cdot\mathbf{D}^{e}\cdot\mathbf{R}^{e})]\cdot\mathbf{R}^{eT}\;\Rightarrow$$

$$\boxed{\overset{o}{\boldsymbol{\sigma}}\cong\hat{\mathbf{L}}^{e}:\mathbf{D}^{e}} \tag{3.9}$$

where: $\hat{L}^{e}_{ijkl}=R^{e}_{im}R^{e}_{jn}R^{e}_{kp}R^{e}_{lq}L^{e}_{mnpq}$

Recalling now that $\boldsymbol{\omega}\cong\mathbf{W}^{*}=\mathbf{W}-\mathbf{W}^{p}$ the co-rotational stress rate in equation (3.7) becomes:

$$\overset{o}{\boldsymbol{\sigma}}\cong\dot{\boldsymbol{\sigma}}+\boldsymbol{\sigma}\cdot(\mathbf{W}-\mathbf{W}^{p})-(\mathbf{W}-\mathbf{W}^{p})\cdot\boldsymbol{\sigma}=\underbrace{\dot{\boldsymbol{\sigma}}+\boldsymbol{\sigma}\cdot\mathbf{W}-\mathbf{W}\cdot\boldsymbol{\sigma}}_{\overset{\triangledown}{\boldsymbol{\sigma}}}-\boldsymbol{\sigma}\cdot\mathbf{W}^{p}+\mathbf{W}^{p}\cdot\boldsymbol{\sigma}\Rightarrow$$

$$\overset{o}{\boldsymbol{\sigma}}\cong\overset{\triangledown}{\boldsymbol{\sigma}}-\boldsymbol{\sigma}\cdot\mathbf{W}^{p}+\mathbf{W}^{p}\cdot\boldsymbol{\sigma} \tag{3.10}$$

Rearranging the terms in equation (3.10) and taking into consideration equation (3.9) we can conclude on the following form for the *Jaumman* stress rate:

$$\boxed{\overset{\nabla}{\boldsymbol{\sigma}} = \hat{\mathbf{L}}^e : \mathbf{D}^e + \boldsymbol{\sigma} \cdot \mathbf{W}^p - \mathbf{W}^p \cdot \boldsymbol{\sigma}} \tag{3.11}$$

Note that in **isotropic materials** $\hat{\mathbf{L}^e} = \mathbf{L}^e$ and $\mathbf{W}^p = 0$ and as a result we can replace $\dot{\mathbf{S}}^e = \mathbf{L}^e : \dot{\mathbf{E}}^{Ge}$ by:

$$\boxed{\overset{\nabla}{\boldsymbol{\sigma}} = \mathbf{L}^e : \mathbf{D}^e} \tag{3.12}$$

So far we have discussed about the equations used in case of elasticity. Thus, we can discuss the equations governing the plastic behavior of metals.

Plastic equations:

We know that in the case of plasticity it is necessary to define a yield function. This yield function can be determined as:

$$\Phi_i\left(\boldsymbol{\Sigma}^e, q_{i\alpha}\right) \cong \Phi_i\left(\mathbf{R}^{eT} \cdot \boldsymbol{\sigma} \cdot \mathbf{R}^e, \mathbf{R}^{eT}\mathbf{R}^e\left[q_{i\alpha}\right]\right) = \Phi(\boldsymbol{\sigma}, q_\alpha) = 0$$

where: $q_\alpha = \mathbf{R}^e[q_{i\alpha}]$

We proceed with the evaluation of the plastic part of the Deformation rate tensor, $\mathbf{D}^p$, and the plastic part of the Spin tensor, $\mathbf{W}^p$:

- $\mathbf{D}^p \cong \mathbf{R}^e \cdot \mathbf{D}_i^p \cdot \mathbf{R}^{eT} = \dot{\lambda}\mathbf{R}^e \cdot \mathbf{N}_i(\boldsymbol{\Sigma}^e, q_{i\alpha}) \cdot \mathbf{R}^{eT} = \dot{\lambda}\mathbf{N}_i\left(\mathbf{R}^e \cdot \boldsymbol{\Sigma}^e \cdot \mathbf{R}^{eT}, \mathbf{R}^e[q_{i\alpha}]\right) \rightarrow$ $\boxed{\mathbf{D}^p \cong \dot{\lambda}\mathbf{N}_i(\boldsymbol{\sigma}, q_{i\alpha})}$
- $\mathbf{W}^p \cong \mathbf{R}^e \cdot \mathbf{W}_i^p \cdot \mathbf{R}^{eT} = \dot{\lambda}\mathbf{R}^e \cdot \boldsymbol{\Omega}_i(\boldsymbol{\Sigma}^e, \boldsymbol{q_{i\alpha}}) \cdot \mathbf{R}^{eT} = \dot{\lambda}\boldsymbol{\Omega}_i(\mathbf{R}^e \cdot \boldsymbol{\Sigma}^e \cdot \mathbf{R}^{eT}, \mathbf{R}^{eT}, \mathbf{R}^e[q_{i\alpha}]) \rightarrow$ $\boxed{\mathbf{W}^p \cong \dot{\lambda}\boldsymbol{\Omega}_i(\boldsymbol{\sigma}, q_{i\alpha})}$

$$\overset{o}{q}_\alpha = \overset{o}{\overline{\mathbf{R}^e[q_{i\alpha}]}} = \mathbf{R}^e[\dot{q_{i\alpha}}]\dot{\lambda}\mathbf{R}^e\left[g_{i\alpha}(\boldsymbol{\Sigma}^e, q_{i\beta})\right] = \dot{\lambda}g_{i\alpha}\left(\mathbf{R}^e \cdot \boldsymbol{\Sigma}^e \cdot \mathbf{R}^{eT}, \mathbf{R}^e[q_{i\beta}]\right) \cong \dot{\lambda}g_{i\alpha}(\boldsymbol{\sigma}, q_\beta) \Rightarrow$$

$$\boxed{\overset{\nabla}{q}_\alpha \cong \dot{\lambda}\left[g_{i\alpha}(\boldsymbol{\sigma}, q_\beta) + A_\alpha(\boldsymbol{\Omega}_i)\right]}$$

where

$$\overset{\nabla}{q}_\alpha = \begin{cases} \dot{q_\alpha}, & \text{if } q_\alpha \text{ is a scalar,} \\ \dot{\mathbf{q}}_\alpha - \mathbf{W} \cdot \dot{\mathbf{q}}_\alpha, & \text{if } \mathbf{q}_\alpha \text{ is a vector,} \\ \dot{\mathbf{q}}_\alpha + \mathbf{q}_\alpha \cdot \mathbf{W} - \mathbf{W} \cdot \mathbf{q}_\alpha, & \text{if } \mathbf{q}_\alpha \text{ is a second order tensor} \end{cases}$$

and

$$\mathcal{A}_\alpha(\boldsymbol{\Omega}_i) = \begin{cases} 0, & \text{if } q_\alpha \text{ is a scalar,} \\ -\boldsymbol{\Omega}_i \cdot \mathbf{q}_\alpha, & \text{if } \mathbf{q}_\alpha \text{ is a vector,} \\ \mathbf{q}_\alpha \cdot \boldsymbol{\Omega}_i - \boldsymbol{\Omega}_i \cdot \mathbf{q}_\alpha, & \text{if } \mathbf{q}_\alpha \text{ is a second order tensor} \end{cases}$$

A brief summary on the elasto-plastic equations as they expressed above is:

$$\boxed{\overset{\nabla}{\boldsymbol{\sigma}} = \hat{\mathbf{L}}^e : \mathbf{D}^e + \boldsymbol{\sigma} \cdot \mathbf{W}^p - \mathbf{W}^p \cdot \boldsymbol{\sigma}}, \tag{3.13}$$

$$\boxed{\Phi_i(\boldsymbol{\sigma}, q_\alpha) = 0} \quad \text{where} \quad \boxed{q_\alpha = \mathbf{R}^e[q_{i\alpha}]}, \tag{3.14}$$

$$\boxed{\mathbf{D}^p = \dot{\lambda}\mathbf{N}_i(\boldsymbol{\sigma}, q_\alpha)}, \tag{3.15}$$

$$\boxed{\mathbf{W}^p = \dot{\lambda}\boldsymbol{\Omega}_i(\boldsymbol{\sigma}, q_\alpha)} \tag{3.16}$$

$$\boxed{\overset{\nabla}{q}_\alpha = \dot{\lambda} g_{i\alpha}(\boldsymbol{\sigma}, q_\beta + \dot{\lambda}\mathcal{A}_\alpha(\boldsymbol{\Omega}_i))} \tag{3.17}$$

Now let us take into account that $\Phi_i(\boldsymbol{\sigma}, q_\alpha)$ is an isotropic function. Mathematically this can be stated as:

$$0 = \dot{\Phi}_i = \frac{\partial \Phi_i}{\partial \boldsymbol{\sigma}} : \dot{\boldsymbol{\sigma}} + \sum_{\alpha=1}^{n} \frac{\partial \Phi_i}{\partial q_\alpha} \dot{q}_\alpha = \frac{\partial \Phi_i}{\partial \boldsymbol{\sigma}} : \overset{o}{\boldsymbol{\sigma}} + \sum_{\alpha=1}^{n} \frac{\partial \Phi_i}{\partial q_\alpha} \overset{o}{q}_\alpha = \frac{\partial \Phi_i}{\partial \boldsymbol{\sigma}} : \overset{\nabla}{\boldsymbol{\sigma}} + \sum_{\alpha=1}^{n} \frac{\partial \Phi_i}{\partial q_\alpha} \overset{\nabla}{q}_\alpha \tag{3.18}$$

Now, if we set

$$\overset{o}{\boldsymbol{\sigma}} \cong \hat{\mathbf{L}}^e : \mathbf{D}^e \tag{3.19}$$

and

$$\overset{\nabla}{\boldsymbol{\sigma}} \cong \hat{\mathbf{L}}^e : \mathbf{D}^e + \boldsymbol{\sigma} \cdot \mathbf{W}^p - \mathbf{W}^p \cdot \boldsymbol{\sigma} \tag{3.20}$$

in equation (3.18) we find that:

$$\frac{\partial \Phi_i}{\partial \boldsymbol{\sigma}} : \hat{\mathbf{L}}^e : (\mathbf{D} - \dot{\lambda}\mathbf{N}_i) + \dot{\lambda} \underbrace{\sum_{n}^{\alpha=1} \frac{\partial \Phi_i}{\partial q_\alpha} g_{i\alpha}}_{-H} =$$

$$\frac{\partial \Phi_i}{\partial \boldsymbol{\sigma}} : \left[\hat{\mathbf{L}}^e : (\mathbf{D} - \dot{\lambda}\mathbf{N}_i) + \dot{\lambda}(\boldsymbol{\sigma}\cdot\boldsymbol{\Omega}_i - \boldsymbol{\Omega}_i\cdot\boldsymbol{\sigma})\right] + \dot{\lambda}\underbrace{\sum_{\alpha=1}^{n}\frac{\partial \Phi_i}{\partial q_\alpha}(g_{i\alpha} + \mathcal{A}_\alpha)}_{-\dot{H}_j} = 0 \Rightarrow$$

$$\dot{\lambda} = \frac{1}{L}\frac{\partial \Phi_i}{\partial \boldsymbol{\sigma}} : \hat{\mathbf{L}}^e : \mathbf{D} \quad \text{where} \quad L = \frac{\partial \Phi}{\partial \boldsymbol{\sigma}} : \hat{\mathbf{L}}^e : \mathbf{N}_i^p + H = \frac{\partial \Phi_i}{\partial \boldsymbol{\sigma}} : \hat{\mathbf{L}}^e : (\mathbf{N}_i - \boldsymbol{\sigma}\cdot\boldsymbol{\Omega}_i + \boldsymbol{\Omega}_i\cdot\boldsymbol{\sigma}) + H_j$$

Therefore, the elastic equations (3.19) and (3.20) can be written as:

1. $\overset{o}{\boldsymbol{\sigma}} = \hat{\mathbf{L}}^e : \mathbf{D} - \hat{\mathbf{L}}^e : \mathbf{D}^p = \hat{\mathbf{L}}^e : \mathbf{D} - \dot{\lambda}\hat{\mathbf{L}}^e : \mathbf{N}_i = \hat{\mathbf{L}}^e : \mathbf{D} - \frac{1}{L}\left(\frac{\partial \Phi_i}{\partial \boldsymbol{\sigma}} : \hat{\mathbf{L}}^e : \mathbf{D}\right)\hat{\mathbf{L}}^e : \mathbf{N}_i$

2. $\overset{\nabla}{\boldsymbol{\sigma}} = \hat{\mathbf{L}}^e : \mathbf{D} - \hat{\mathbf{L}}^e : \mathbf{D}^p + \boldsymbol{\sigma}\cdot\mathbf{W}^p - \mathbf{W}^p\cdot\boldsymbol{\sigma} = \hat{\mathbf{L}}^e : \mathbf{D} - \dot{\lambda}\left(\hat{\mathbf{L}}^e : \mathbf{N}_i - \boldsymbol{\sigma}\cdot\boldsymbol{\Omega}_i + \boldsymbol{\Omega}_i\cdot\boldsymbol{\sigma}\right) \Rightarrow$
   $\overset{\nabla}{\boldsymbol{\sigma}} = \hat{\mathbf{L}}^e : \mathbf{D} - \frac{1}{L}\left(\frac{\partial \Phi_i}{\partial \boldsymbol{\sigma}} : \hat{\mathbf{L}}^e : \mathbf{D}\right)(\hat{\mathbf{L}}^e : \mathbf{N}_i - \boldsymbol{\sigma}\cdot\boldsymbol{\Omega}_i + \boldsymbol{\Omega}_i\cdot\boldsymbol{\sigma})$

3.2 Numerical Integration

3.2.i Numerical integration schemes - Forward and Backward Euler

There is a variety of numerical methods that can be used in order to solve differential equations. Forward and Backward Euler are two methods that are commonly deployed to solve differential equations associated with finite element analysis because their implementation is simple and their computational cost low. Using a first order ODE as an example we have

$$y_t = f(t, y), \quad y(t_0) = y_0 \tag{3.21}$$

where f is a given smooth function

- **Forward Euler**:
  The Forward Euler's method is an explicit method. Explicit methods calculate the state of the system at a later time from the state of the system at the current time without the need to solve algebraic equations. For the Forward method, we begin by choosing a step size or Δt. The size of Δt determines the accuracy of the approximate solutions as well as the number of computations.
  Let t_n, n=0,1,2,.., be a sequence in time with

$$t_{n+1} = t_n + \Delta t \tag{3.22}$$

Also, let y_n and Y_n be the exact and the approximate solution at $t = t_n$, respectively. To obtain Y_{n+1} from (t_n, Y_n), we use the differential equation (3.21). Since the slope of the solution to the equation $y_n = f(t, y)$ at the point (t_n, y_n) is $f(t_n, y_n)$, the Euler method determines the point (t_{n+1}, Y_{n+1}) by assuming that it lies on the line through (t_n, Y_n) with the slope $f(t_n, Y_n)$. Hence, the formula for the slope of a line gives:

$$\frac{Y_{n+1} - Y_n}{\Delta t} = f(t_n, Y_n) \tag{3.23}$$

or

$$Y_{n+1} = Y_n + f(t_n, Y_n)\Delta t \tag{3.24}$$

As the step size decreases then the error between the actual and the approximate solution is reduced.

- **Backward Euler**:
  Backward Euler is an implicit method that finds the solution by solving an equation involving the current state of the system and the later one. More precisely,

$$Y_{n+1} = Y_n + f(t_n, Y_{n+1}\Delta t) \tag{3.25}$$

The stability that this numerical method has, is its main advantage over the Forward Euler. On the other hand, it is not as computationally efficient as the Forward Euler since a system of equations needs to be solved every time step.

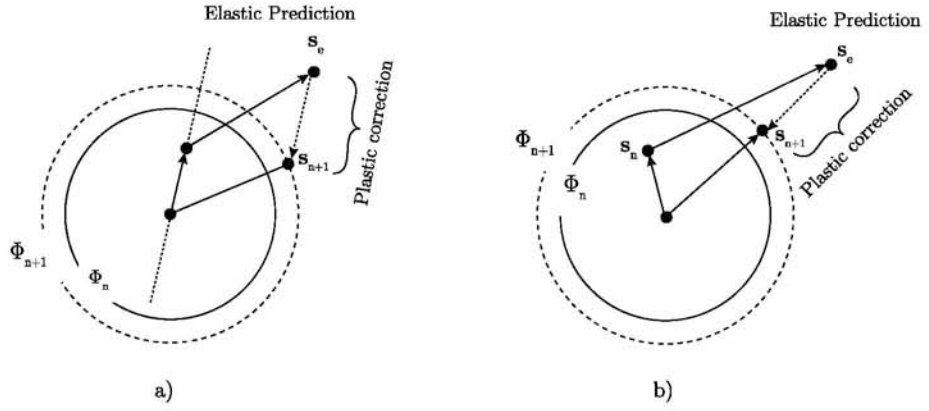


*Fig. 3.1: A schematic representation of the stress integration shown in the stress domain for isotropic hardening with (a)the Forward Euler and (b)the Backward Euler(also called **radial return**)*

3.2.ii Numerical Integration of the FEA analysis

We have already declared which are the constitutive equations governing elasticity and plasticity. Now, it is time to proceed with the numerical integration of those equations. Let us begin by writing the constitutive equations:

$$\begin{aligned}
&\mathbf{D} = \mathbf{D}^e + \mathbf{D}^p, \quad \overset{\nabla}{\boldsymbol{\sigma}} = \hat{\mathbf{L}}^e : \mathbf{D}^e + \boldsymbol{\sigma}\cdot\mathbf{W}^p - \mathbf{W}^p\cdot\boldsymbol{\sigma} \\
&\mathbf{D}^p = \dot{\lambda}\mathbf{N}(\boldsymbol{\sigma}, q_\alpha), \quad \mathbf{W}^p = \dot{\lambda}\boldsymbol{\Omega}(\boldsymbol{\sigma}, q_\alpha) \\
&\Phi(\boldsymbol{\sigma}, q_\alpha) = 0, \quad \overset{\nabla}{q}_\alpha = \dot{\lambda} f_\alpha(\boldsymbol{\sigma}, q_\beta)
\end{aligned} \tag{3.26}$$

where Φ, $\mathbf{N}$, $\boldsymbol{\Omega}$ and f_α are isotropic functions.

The computational problem within every integration point of each finite element can be formulated as:

$$\boxed{\textbf{Given: } \boldsymbol{\sigma}_n,\ \mathbf{F}_n,\ q_\alpha|_n\ ,\ \mathbf{F}_{n+1}} \quad \longrightarrow \quad \boxed{\textbf{Find: } \quad \boldsymbol{\sigma}_{n+1},\ q_\alpha|_{n+1}}$$

During this time increment the deformation gradient can be written in the form:

$$\mathbf{F}(t) = \Delta\mathbf{F}(t)\cdot\mathbf{F}_n = \mathbf{R}(t)\cdot\mathbf{U}(t)\cdot\mathbf{F}_n, \qquad t_n \le t \le t_{n+1} \tag{3.27}$$

where $\mathbf{R}(t)$ and $\mathbf{U}(t)$ are the rotation and the right stretch tensors associated with $\Delta\mathbf{F}(t)$. The corresponding deformation rate $\mathbf{D}(t)$ tensor and the spin tensor $\mathbf{W}(t)$ can be written as:

$$\mathbf{D}(t) \equiv \left[\mathbf{F}(t)\dot{\cdot}\,\mathbf{F}^{-1}\right]_{symm} = \left[\Delta\dot{\mathbf{F}}(t)\cdot\Delta\mathbf{F}^{-1}(t)\right]_{symm}, \tag{3.28}$$

$$\mathbf{W}(t) \equiv \left[\mathbf{F}(t)\dot{\cdot}\,\mathbf{F}^{-1}\right]_{antisymm} = \left[\Delta\dot{\mathbf{F}}(t)\cdot\Delta\mathbf{F}^{-1}(t)\right]_{antisymm} \tag{3.29}$$

If it is assumed that the Lagrangian triad associated with $\Delta\mathbf{F}(t)$ remains fixed in the time interval $[t_n, t_{n+1}]$, it can readily be shown that:

$$\mathbf{D}(t) = \mathbf{R}(t)\cdot\dot{\mathbf{E}}(t)\cdot\mathbf{R}^T(t), \quad \text{and} \quad \mathbf{W}(t) = \dot{\mathbf{R}}(t)\cdot\mathbf{R}^T \tag{3.30}$$

whereas

$$\overset{\nabla}{\boldsymbol{\sigma}} = \mathbf{R}(t) \cdot \dot{\hat{\boldsymbol{\sigma}}} \cdot \mathbf{R}^T(t), \qquad \overset{\nabla}{q_\alpha} = \mathbf{R}\left[\dot{\hat{q}}_\alpha\right] \tag{3.31}$$

where $\mathbf{E}(t) = \ln \mathbf{U}(t)$ is the logarithmic strain associated with the increment, and

$$\hat{\boldsymbol{\sigma}}(t) = \mathbf{R}^T(t) \cdot \boldsymbol{\sigma}(t) \cdot \mathbf{R}(t) \tag{3.32}$$

$$\boxed{\textbf{Start of the incremet } (t = t_n) : \Delta\mathbf{F}_n = \mathbf{R}_n = \mathbf{U}_n = \boldsymbol{\delta}, \quad \hat{\boldsymbol{\sigma}}_n = \boldsymbol{\sigma}_n \quad \text{and} \quad \mathbf{E}_n = 0}$$

$$\Big\downarrow$$

$$\boxed{\textbf{End of the incremet } (t = t_{n+1}) : \Delta\mathbf{F}_{n+1} = \mathbf{F}_{n+1} \cdot \mathbf{F}_n^{-1} = \mathbf{R}_{n+1} \cdot \mathbf{U}_{n+1}, \quad \mathbf{E}_{n+1} = \ln \mathbf{U}_{n+1}}$$

Note that at the end of the increment $(t = t_{n+1})$ the quantities $\Delta\mathbf{F}_{n+1}$, $\mathbf{E}_{n+1}$ are known.

Now we proceed by modifying the constitutive equations (3.26):

- $\mathbf{D} = \mathbf{D}^e + \mathbf{D}^p \quad \Rightarrow \quad \mathbf{R} \cdot \dot{\mathbf{E}} \cdot \mathbf{R}^T = \mathbf{R} \cdot \dot{\mathbf{E}}^e \cdot \mathbf{R}^T + \mathbf{R} \cdot \dot{\mathbf{E}}^p \cdot \mathbf{R}^T \quad \Rightarrow \quad \boxed{\dot{\mathbf{E}} = \dot{\mathbf{E}}^e + \dot{\mathbf{E}}^p}$

- $\overset{\nabla}{\boldsymbol{\sigma}} = \hat{\mathbf{L}}^e : \mathbf{D}^e + \boldsymbol{\sigma} \cdot \mathbf{W}^p - \mathbf{W}^p \cdot \boldsymbol{\sigma} \quad \Rightarrow$
  $\mathbf{R} \cdot \dot{\hat{\boldsymbol{\sigma}}} \cdot \mathbf{R}^T = \hat{\mathbf{L}}^e : \left(\mathbf{R} \cdot \dot{\mathbf{E}}^e \cdot \mathbf{R}^T\right) + \boldsymbol{\sigma} \cdot \mathbf{W}^p - \mathbf{W}^p \cdot \boldsymbol{\sigma} \Rightarrow$
  $\dot{\hat{\boldsymbol{\sigma}}} = \mathbf{R}^T \cdot \left[\hat{\mathbf{L}}^e : \left(\mathbf{R} \cdot \dot{\mathbf{E}}^e \cdot \mathbf{R}^T\right)\right] \cdot \mathbf{R} +$
  $+\mathbf{R}^T \cdot \boldsymbol{\sigma} \cdot \mathbf{R} \cdot \mathbf{R}^T \cdot \mathbf{W}^p(\boldsymbol{\sigma}, q_\alpha) \cdot \mathbf{R} - \mathbf{R}^T \cdot \mathbf{W}^p(\boldsymbol{\sigma}, q_\alpha) \cdot \mathbf{R} \cdot \mathbf{R}^T \cdot \boldsymbol{\sigma} \cdot \mathbf{R} \Rightarrow$
  $\boxed{\dot{\hat{\boldsymbol{\sigma}}} = \tilde{\mathbf{L}}^e : \mathbf{E}^e + \dot{\lambda}\hat{\boldsymbol{\sigma}} \cdot \boldsymbol{\Omega}^p(\hat{\boldsymbol{\sigma}}, \hat{q}_\alpha) - \dot{\lambda}\boldsymbol{\Omega}^p(\hat{\boldsymbol{\sigma}}, \hat{q}_\alpha) \cdot \hat{\boldsymbol{\sigma}}}$,
  where: $\bar{L}^e_{ijkl} = R_{mi} R_{nj} R_{pk} R_{ql} \hat{L}^e_{mnpq}$

- $\mathbf{D}^p = \dot{\lambda}\mathbf{N}(\boldsymbol{\sigma}, q_\alpha) \;\Rightarrow\; \mathbf{R} \cdot \dot{\mathbf{E}}^p \cdot \mathbf{R}^T = \dot{\lambda}\mathbf{N}(\boldsymbol{\sigma}, q_\alpha) \;\Rightarrow\; \dot{\mathbf{E}}^p = \dot{\lambda}\mathbf{R}^T \cdot \mathbf{N}(\boldsymbol{\sigma}, q_\alpha) \cdot \mathbf{R} \;\Rightarrow$
  $\boxed{\dot{\mathbf{E}}^p = \dot{\lambda}\mathbf{N}(\hat{\boldsymbol{\sigma}}, \hat{q}_\alpha)}$ where: $\hat{q}_\alpha = \mathbf{R}^T[q_\alpha]$

- $\Phi(\boldsymbol{\sigma}, q_\alpha) = 0 \;\Rightarrow\; \Phi(\mathbf{R} \cdot \hat{\boldsymbol{\sigma}} \cdot \mathbf{R}^T, \mathbf{R}[\hat{q}_\alpha]) = 0 \;\Rightarrow\; \boxed{\Phi(\hat{\boldsymbol{\sigma}}, \hat{q}_\alpha) = 0}$

- $\overset{\nabla}{q}_\alpha = \dot{\lambda} f_\alpha(\boldsymbol{\sigma}, q_\beta) \;\Rightarrow\; \mathbf{R}[\dot{\hat{q}}_\alpha] = \dot{\lambda}\mathbf{R}^T\left[f_\alpha(\boldsymbol{\sigma}, q_\beta)\right] \;\Rightarrow\; \boxed{\dot{\hat{q}}_\alpha = \dot{\lambda} f_\alpha(\hat{\boldsymbol{\sigma}}, \hat{q}_\beta)}$

We will use the Backward-Euler scheme to integrate numerically the flow rule $\dot{\mathbf{E}} = \dot{\lambda}\mathbf{N}(\hat{\boldsymbol{\sigma}}, \hat{q}_\alpha)$ and the Forward Euler scheme to integrate numerically the quantities $\dot{\hat{\boldsymbol{\sigma}}}$ and $\dot{\hat{q}}_\alpha$. We discussed about those numerical integration schemes in section 3.2.i.

- $\Delta\mathbf{E} = \Delta\mathbf{E}^e + \Delta\mathbf{E}^p,$

- $\hat{\boldsymbol{\sigma}}_{n+1}\left(\Delta\mathbf{E}^p, \Delta\lambda\right) = \underbrace{\hat{\boldsymbol{\sigma}}_n + \hat{\mathbf{L}}^e_n : \Delta\mathbf{E}}_{\boldsymbol{\sigma}^e=\text{known}} - \mathbf{L}^e_n : \Delta\mathbf{E}^p + \Delta\lambda\left(\hat{\boldsymbol{\sigma}}_n \cdot \boldsymbol{\Omega}^p_n - \boldsymbol{\Omega}^p_n \cdot \hat{\boldsymbol{\sigma}}_n\right)$ (3.33)

- $\Delta\mathbf{E}^p = \Delta\lambda\mathbf{N}\left(\hat{\boldsymbol{\sigma}}_{n+1}\left(\Delta\mathbf{E}^p, \Delta\lambda\right), \hat{q}_\alpha|_{n+1}\left(\Delta\lambda\right)\right),$

- $\Phi\left(\hat{\boldsymbol{\sigma}}_{n+1}\left(\Delta\mathbf{E}^p, \Delta\lambda\right), \hat{q}_\alpha|_{n+1}(\Delta\lambda)\right) = 0,$

- $\Delta\hat{q}_\alpha(\Delta\lambda) = \Delta\lambda f_\alpha(\hat{\boldsymbol{\sigma}}_n, \hat{q}_\alpha|_n)$ (3.34)

where we took into account that $\bar{\mathbf{L}}^e_n = \hat{\mathbf{L}}^e_n$

We consider $\Delta\mathbf{E}^p$ and $\Delta\lambda$ as the primary unknowns with

$$\Delta\mathbf{E}^p - \Delta\lambda\mathbf{N}\left(\hat{\boldsymbol{\sigma}}_{n+1}\left(\Delta\mathbf{E}^p, \Delta\lambda\right), \hat{q}_\alpha|_{n+1}(\Delta\lambda)\right) = 0, \quad \text{and} \tag{3.35}$$

$$\Phi\left(\hat{\boldsymbol{\sigma}}_{n+1}\left(\Delta\mathbf{E}^p, \Delta\lambda\right), \hat{q}_\alpha|_{n+1}(\Delta\lambda)\right) = 0 \tag{3.36}$$

as the basic equations. The non-linear equations (3.35) and (3.36) are solved for $\Delta\mathbf{E}^p$ and $\Delta\lambda$ where $\hat{\boldsymbol{\sigma}}_{n+1}$ and $\hat{q}_\alpha|_{n+1}$ are determined from equations (3.33) and (3.34) respectively. Finally, we can calculate the quantities $\boldsymbol{\sigma}_{n+1}$ and $q_\alpha|_{n+1}$ from equations:

$$\boxed{\boldsymbol{\sigma}_{n+1} = \mathbf{R}_{n+1} \cdot \hat{\boldsymbol{\sigma}}_{n+1} \cdot \mathbf{R}^T_{n+1}} \quad \text{and} \quad \boxed{q_\alpha|_{n+1} = \mathbf{R}_{n+1}\left[\hat{q}_\alpha|_{n+1}\right]}$$

Chapter 4

Finite Element Formulation

4.1 Introduction

Let us consider a general continuum body in the reference configuration (t=0), which occupies volume V_0 with a mass density ρ_0. The body is then loaded by body forces $\boldsymbol{b}$ per unit mass and traction forces $\boldsymbol{t}$ acting on the surrounding surface S_t, while another part of the surrounding surface called S_u is subjected to known displacements $\hat{\boldsymbol{u}}$. After Δt period of time, the body is deformed and occupies volume V with a mass density ρ, surrounded by surface S. Thus the equilibrium equations in terms of the Cauchy stress tensor can be expressed as follows:

$$\frac{\partial \sigma_{ij}}{\partial x_j} + \rho b_i = 0 \tag{4.1}$$

We introduce the kinematic relationships:

$$D_{ij} = \frac{1}{2}\left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right) \tag{4.2}$$

where $\boldsymbol{D}$ is the deformation rate tensor and $\boldsymbol{v}$ is the velocity field. We also introduce the conditions defining the applied forces and displacements in the boundary ∂S:

$$\mathbf{u} = \hat{\mathbf{u}} = \text{known on } S_u \tag{4.3}$$

$$\hat{\boldsymbol{t}} = \boldsymbol{\sigma} \cdot \mathbf{n} = \text{known on } S_t \tag{4.4}$$

Also, we define a constitutive law for the deformable material of the form:

$$\boldsymbol{\sigma} = \boldsymbol{\sigma}(\mathbf{E}) \tag{4.5}$$

where $\mathbf{E}$ is the logarithmic strain.

The above equations (4.1)-(4.3) constitute the **Strong Formulation** of the Boundary Value Problem. Now, we will the **Weak Formulation** of the Boundary Value Problem (or BVP) which provides the basis of the **Finite Element approximation**. To begin with, we replace the three equilibrium equations in (4.1) by a unique scalar equation [1] over the entire body. Then, we multiply the differential equation in (4.3) by a virtual (arbitrary but continuous and differentiable) velocity field $\delta\boldsymbol{v}^*$ and then integrating over the entire volume of the continuum body. Thus, we have the following equation:

$$\int_{V(t)} [\boldsymbol{\nabla} \cdot \boldsymbol{\sigma} + \rho \boldsymbol{b}] \cdot \boldsymbol{v}^* dV = 0 \tag{4.6}$$

Now making use of the chain rule to write:

$$\boldsymbol{\nabla} \cdot (\boldsymbol{\sigma} \cdot \boldsymbol{v}^*) = (\boldsymbol{\nabla} \cdot \boldsymbol{\sigma}) \cdot \boldsymbol{v}^* + \boldsymbol{\sigma} : (\boldsymbol{\nabla} \boldsymbol{v}^*)$$

and using Gauss's theorem we can write the following:

$$\begin{aligned} \int_{V(t)} [\boldsymbol{\nabla} \cdot \boldsymbol{\sigma}] \cdot \boldsymbol{v}^* dV &= \int_{V(t)} [\boldsymbol{\nabla} \cdot (\boldsymbol{\sigma} \cdot \boldsymbol{v}^*) - \boldsymbol{\sigma} : (\boldsymbol{\nabla} \boldsymbol{v}^*)] \, dV \\ &= \int_{S(t)} \mathbf{n} \cdot \boldsymbol{\sigma} \cdot \boldsymbol{v}^* dS - \int_{V(t)} \boldsymbol{\sigma} : (\boldsymbol{\nabla} \boldsymbol{v}^*) \\ &= \int_{S(t)} \hat{\boldsymbol{t}} \cdot \boldsymbol{v}^* dS - \int_{V} \boldsymbol{\sigma} : \boldsymbol{L}^* dV \end{aligned} \tag{4.7}$$

where $\boldsymbol{L}^*$ is the velocity gradient tensor corresponding to the virtual velocity field $\delta\boldsymbol{v}^*$. We know that $\delta\boldsymbol{L}^*$ can be decomposed into its symmetric $\boldsymbol{D}^*$ and antisymmetric part $\boldsymbol{W}^*$, thus we are able to write the following:

[1] this replacement does not violate the generality

$$\boldsymbol{\sigma} : \boldsymbol{L}^* = \boldsymbol{\sigma} : (\boldsymbol{D}^* + \boldsymbol{W}^*) = \boldsymbol{\sigma} : \boldsymbol{D}^* + \boldsymbol{\sigma} : \boldsymbol{W}^* = \boldsymbol{\sigma} : \boldsymbol{D}^* \tag{4.8}$$

Note that we took advantage of symmetry of $\boldsymbol{\sigma}$ to write $\boldsymbol{\sigma} : \boldsymbol{L}^* = \boldsymbol{\sigma} : \boldsymbol{D}^*$ as we know that the double dot product of a symmetric and an antisymmetric tensor equals to zero.

Now combining equations (4.6), (4.7) and (4.8) we can conclude on the alterative formulation of the BVP:

$$\boxed{G(\boldsymbol{v}) = \int_{V(t)} \boldsymbol{\sigma} : \boldsymbol{D}^* dV - \int_{S(t)} \hat{\boldsymbol{t}} \cdot \boldsymbol{v}^* dS - \int_{V(t)} \rho \boldsymbol{b} \cdot \boldsymbol{v}^* dV = 0} \tag{4.9}$$

which is the so called **Weak Formulation** and provides the basis for the **Finite Element approximation** which we will introduce in the following section.

4.2 Finite Element Approximation

In a finite element setting, the problem is solved incrementally and the primary unknown is the displacement increment $\Delta\mathbf{u}(\mathbf{x})$ that defines the position of the body at the end of an increment:

$$\mathbf{u}_{n+1}(\mathbf{x}) = \mathbf{u}_n(\mathbf{x}) + \Delta\mathbf{u}(\mathbf{x}) \tag{4.10}$$

Then, the current position of any material point within the continuum body can be directly updated as:

$$\mathbf{x}_{n+1}(\mathbf{x}) = \mathbf{x}_n(\mathbf{x}) + \Delta\mathbf{u}(\mathbf{x}) = \mathbf{X} + \mathbf{u}_{n+1}(\mathbf{x}) \tag{4.11}$$

Discretizing the continuum body into finite elements, we express the unknown displacement increment $\Delta\mathbf{u}$ as a function interpolation within each element as:

$$\underset{3\times 1}{\{\Delta u(\mathbf{x})\}} = \underset{3\times n}{[N_u(\mathbf{x})]} \underset{n\times 1}{\{\Delta u^N\}} \tag{4.12}$$

where $[N_u(\mathbf{x})]$ is the interpolation matrix that consists of user-defined "shape" functions and $\{\Delta u^N\}$ is the vector of nodal unknowns. Since the virtual velocity field, $\delta\boldsymbol{v}^*$ must

be compatible with all kinematic constraints and the interpolation introduced in (4.12), constraints the displacement to have a certain spatial variation, then $\delta \boldsymbol{v}^*$ must also be defined using the same function interpolation. Thus,

$$\underset{3\times 1}{\{\delta v^*\}} = \underset{3\times N}{[N(\mathbf{x})]}\, \underset{n\times 1}{\Delta v^{*N}} \tag{4.13}$$

and the virtual velocity tensor $\delta \mathbf{D}^*$ is also expressed in array form as:

$$\underset{6\times 1}{\{\delta D^*\}} = \underset{6\times n}{[B(\mathbf{x})]}\, \underset{n\times 1}{\{\Delta v^{*N}\}} \tag{4.14}$$

where the matrix $[B(\mathbf{x})]$ containing the spatial derivatives of the shape functions $N(\mathbf{x})$. The following notations are used in the rest of our analysis:

- the Cauchy stress, $\boldsymbol{\sigma}$, is considered as: $\boldsymbol{\sigma} \rightarrow \underset{6\times 1}{\{\sigma\}}$
- the traction forces, $\hat{\boldsymbol{t}}$, are considered as: $\hat{\boldsymbol{t}} \rightarrow \underset{3\times 1}{\{t\}}$
- and the body forces, $\boldsymbol{b}$, are considered as: $\boldsymbol{b} \rightarrow \underset{3\times 1}{\{b\}}$

Now, substituting each term into the **Weak Formulation** in (4.9), we derive the following equation:

$$[\Delta v_e^{*N}] \underset{e}{\mathbf{A}} \left[\int\limits_{V_{n+1}^e} \left([B]_{n+1}^T \{\sigma\}_{n+1} - [N]_{n+1}^T \{b\}_{n+1} \right) dV^e - \int\limits_{S_{n+1}^e} [N]_{n+1}^T \{t\}_{n+1} dS^e \right] = 0 \overset{\forall [\Delta v_e^{*N}]}{\rightarrow}$$

$$\underset{e}{\mathbf{A}} \left[\int\limits_{V_{n+1}^e} [B]_{n+1}^T \{\sigma\}_{n+1} dV^e - \int\limits_{S_{n+1}^e} [N]_{n+1}^T \{b\}_{n+1} dV^e - \int\limits_{S_{n+1}^e} [N]_{n+1}^T \{t\}_{n+1} dS^e \right] = 0$$

The last two integrals comprising of the traction and body forces in the above expression, define the external load vector as:

$$\{F\}_{n+1}^{ext} = \underset{e}{\mathbf{A}} \left[\int\limits_{S_{n+1}^e} [N]_{n+1}^T \{b\}_{n+1} dV^e + \int\limits_{S_{n+1}^e} [N]_{n+1}^T \{t\}_{n+1} dS^e \right] = 0$$

Finally, we can define the residual forces vector which essentially expresses the difference between the internal σ_{n+1} and external t_{n+1}, b_{n+1} forces:

$$\boxed{\{R(\Delta u^N)\}_{n+1} \equiv \mathop{\mathbf{A}}_{e} \int_{V^e_{n+1}} [B]^T_{n+1}\{\sigma\}_{n+1} dV^e - \{F\}^{ext}_{n+1} = \{0\}} \tag{4.15}$$

where $\{\sigma\}_{n+1}$ is usually a non-linear function of the unknowns $\{\Delta u^N\}$.

The solution of the 'weak' formulation (4.9) is the displacement field $\{\Delta u^N\}$ that satisfies the system of equations in equation (4.15), or equivalently, the displacement field that at $t = t_{n+1}$ equates the applied loads $\{F\}^{ext}_{n+1}$ to the internal forces $\{\sigma\}_{n+1}$.

4.3 Calculation of the Jacobian

Now we will try to derive an expression for the Jacobian $[K]$ from the continuum form in (4.9) by calculating the quantity dG and then introduce the finite element approximation. To begin with, we will express all integrals involved in (4.9) with respect to the reference configuration, in order to avoid variations of the form $V=V(t)$ and $S=S(t)$, involved in the limits of integration, when taking the derivative dG. Furthermore, we will take into account equation (4.8) so that to replace $\boldsymbol{\sigma} : \delta \boldsymbol{D}^*$ with $\boldsymbol{\sigma} : \delta \boldsymbol{L}^*$:

$$G(\Delta \boldsymbol{u}) = \int_V tr(\boldsymbol{\sigma} \cdot \delta \boldsymbol{L}^*) dV \; - \int_{V_0} \rho_0 \boldsymbol{b} \cdot \delta \boldsymbol{v}^* dV_0 \; - \int_{S_0} \hat{\boldsymbol{t}}_0 \cdot \delta \boldsymbol{v}^* dS_0 \tag{4.16}$$

where ρ_0 and $\hat{\boldsymbol{t}}_0$ stand for the mass density and the normal traction vector at $t=0$ respectively. Recalling equations (4.10) and (4.11) we note that:

$$d\mathbf{x}_{n+1} = d(\mathbf{X} + \mathbf{u}_{n+1}) = d(\mathbf{x}_n + \Delta \mathbf{u}) = d(\Delta \mathbf{u}) \tag{4.17}$$

Now we rewrite the first integral in (4.16) from the current to the reference state.

$$\int_V tr(\boldsymbol{\sigma} \cdot \delta \boldsymbol{L}^*) dV = \int_{V_0} tr\left(\boldsymbol{\sigma} \cdot \frac{\partial \delta \boldsymbol{v}^*}{\partial \mathbf{X}} \cdot \frac{\partial \mathbf{X}}{\partial \mathbf{x}}\right) J \, dV_0 = \int_{V_0} tr\left(\boldsymbol{\sigma} \cdot \frac{\partial \delta \boldsymbol{v}^*}{\partial \mathbf{X}} \cdot \mathbf{F}^{-1}\right) J dV_0$$

Substituting into (4.16), the Weak formulation of the BVP can be written with respect to the reference state as:

$$G(\Delta \mathbf{u}) = \int_{V_0} tr\left(\frac{\partial \delta \boldsymbol{v}^*}{\partial \mathbf{X}} \cdot \mathbf{F}^{-1} \cdot \boldsymbol{\sigma}\right) J dV_0 \; - \int_{V_0} \rho_0 \boldsymbol{b} \cdot \delta \boldsymbol{v}^* dV_0 \; - \int_{S_0} \hat{\boldsymbol{t}}_0 \cdot \delta \boldsymbol{v}^* dS_0 \tag{4.18}$$

Assuming that all initially applied forces are independent of the body's motion, we can derive the following:

$$dG = \int_{V_0} tr\left[\frac{\partial \delta \boldsymbol{v}^*}{\partial \mathbf{X}} \cdot \left(d\mathbf{F}^{-1} \cdot \boldsymbol{\sigma} + \mathbf{F}^{-1} \cdot d\boldsymbol{\sigma} + \mathbf{F}^{-1} \cdot \boldsymbol{\sigma}\frac{dJ}{J}\right)\right] JdV_0 \tag{4.19}$$

Recalling that: $\dfrac{\partial \delta \boldsymbol{v}^*}{\partial \mathbf{X}} = \dfrac{\partial \delta \boldsymbol{v}^*}{\partial \mathbf{x}} \cdot \dfrac{\partial \mathbf{x}}{\partial \mathbf{X}} = d\mathbf{L}^* \cdot \mathbf{F}$ the expression for dG becomes:

$$dG = \int_{V_0} tr\left[\delta \boldsymbol{L}^* \cdot \left(\mathbf{F} \cdot d(\mathbf{F}^{-1}) \cdot \boldsymbol{\sigma} + d\boldsymbol{\sigma} + \boldsymbol{\sigma}\frac{dJ}{J}\right)\right] JdV_0 \tag{4.20}$$

At this point we only need to evaluate the expressions $\mathbf{F} \cdot d(\mathbf{F}^{-1})$, $d\boldsymbol{\sigma}$ and dJ/J.

- Evaluation of $\mathbf{F} \cdot d(\mathbf{F}^{-1})$:

  $$\mathbf{F} \cdot \mathbf{F}^{-1} = \boldsymbol{\delta} \;\Rightarrow\; d\mathbf{F} \cdot \mathbf{F}^{-1} + \mathbf{F} \cdot d(\mathbf{F}^{-1}) = 0 \;\Rightarrow\; \mathbf{F} \cdot d(\mathbf{F}^{-1}) = -d\mathbf{F} \cdot \mathbf{F}^{-1}$$

  where

  $$d\mathbf{F} = d\left(\frac{\partial \mathbf{x}}{\partial \mathbf{X}}\right) = \frac{\partial (d\mathbf{x})}{\partial \mathbf{X}} = \frac{\partial (d\Delta\mathbf{u})}{\partial \mathbf{X}} \tag{4.21}$$

  Now combining the above two expressions we have:

  $$\mathbf{F} \cdot d(\mathbf{F}^{-1}) = -\frac{\partial (d(\Delta\mathbf{u}))}{\partial \mathbf{x}} \;\Rightarrow$$
  $$\mathbf{F} \cdot d(\mathbf{F}^{-1}) = -d\mathbf{L} \tag{4.22}$$

- Evaluation of $\frac{dJ}{J}$:

  Recalling the definition of J as the determinant of the deformation gradient $\mathbf{F}$ we write:

  $$J = det\mathbf{F}$$

  Using Jacobi's formula, we can express the quantity dJ(the derivative of the determinant of $\mathbf{F}$) in terms of the adjugate matrix of $\mathbf{F}$ and d$\mathbf{F}$ as follows:

  $$dJ = tr[adj(\mathbf{F}) \cdot d\mathbf{F}]$$

  and since the inverse of $\mathbf{F}^{-1}$ exists, the adjugate of $\mathbf{F}$ is given by:

  $$adj(\mathbf{F}) = J\mathbf{F}^{-1}$$

and thus:

$$dJ = Jtr[\mathbf{F}^{-1} \cdot d\mathbf{F}]$$

We have already shown that:

$$d\mathbf{F} \cdot \mathbf{F}^{-1} = d\mathbf{L}$$

Therefore, combining the last two expressions, the quantity $\mathrm{d}J/J$ can be expressed as:

$$\frac{dJ}{J} = dL_{kk} \tag{4.23}$$

We will now proceed to the substitution of the expressions in equations (4.23) and (4.22) into equation (4.20) for dG, taking also into account that $tr[\mathbf{A} \cdot \mathbf{B} = \mathbf{A} : \mathbf{B}^T]$:

$$dG = \int_{V(t)} \delta\mathbf{L}^* : [d\boldsymbol{\sigma} - \boldsymbol{\sigma} \cdot d\mathbf{L}^T + \boldsymbol{\sigma} dL_{kk}]dV \tag{4.24}$$

- Evaluation of $\mathrm{d}\boldsymbol{\sigma}$:

  The quantity $\mathrm{d}\boldsymbol{\sigma}$ expresses the stress variation with respect to the displacement increment $\Delta\mathbf{u}$.

  First, we will express the time variation of the deformation gradient F during the time increment $[t_n, t_{n+1}]$ as:

$$\mathbf{F}(t) = \Delta\mathbf{F}(t) \cdot \mathbf{F}_n \tag{4.25}$$

  and the corresponding deformation rate tensor $\mathbf{D}$ can be expressed as:

$$\mathbf{D}(t) = \left[\dot{\mathbf{F}}(t) \cdot \mathbf{F}^{-1}\right]_s = \left[\Delta\dot{\mathbf{F}}(t) \cdot \Delta\mathbf{F}^{-1}\right] \tag{4.26}$$

  Using Polar Decomposition theorem, we can write:

$$\Delta\mathbf{F}(t) = \mathbf{R}(t) \cdot \mathbf{U}(t) \tag{4.27}$$

  and assuming that the Lagrangian triad associated with $\Delta\mathbf{F}(t)$ remains fixed over the period of one increment we can substitute the polar decomposition theorem of $\Delta\mathbf{F}(t)$ to (4.26) and write $\mathbf{D}$ in the form:

$$\mathbf{D}(t) = \mathbf{R}(t) \cdot \dot{\mathbf{E}}(t) \cdot \mathbf{R}^T(t) \tag{4.28}$$

  where $\mathbf{E}$ stands for the logarithmic strain. Now let us recall the Jaumman rate of Cauchy stress which is given by:

$$\overset{\nabla}{\boldsymbol{\sigma}} = \dot{\boldsymbol{\sigma}} - \mathbf{W} \cdot \boldsymbol{\sigma} + \boldsymbol{\sigma} \cdot \mathbf{W} \tag{4.29}$$

It can easily be shown that the spin tensor $\mathbf{W}(t)$ is in this case given by:

$$\mathbf{W}(t) = \dot{\mathbf{R}}(t) \cdot \mathbf{R}^T(t) \tag{4.30}$$

and thus the Jaumman rate of Cauchy stress is expressed as:

$$\overset{\nabla}{\boldsymbol{\sigma}} = \mathbf{R}(t) \cdot \frac{d}{dt}[\hat{\boldsymbol{\sigma}}(t) \cdot \mathbf{R}^T(t)] \quad \text{where:} \quad \hat{\boldsymbol{\sigma}}(t) = \mathbf{R}^T(t) \cdot \boldsymbol{\sigma}(t) \cdot \mathbf{R}(t) \tag{4.31}$$

As far as the quantity $\dot{\hat{\boldsymbol{\sigma}}}(t)$ is concerned we note that:

$$\dot{\hat{\boldsymbol{\sigma}}}(t) = \frac{\partial \hat{\boldsymbol{\sigma}}(t)}{\partial \mathbf{E}(t)} : \dot{\mathbf{E}}(t) = \hat{\mathbf{C}}(t) : \left(\mathbf{R}^T \cdot \mathbf{D}(t) \cdot \mathbf{R}(t)\right) \tag{4.32}$$

Now using the final result for the quantity $\dot{\hat{\boldsymbol{\sigma}}}(t)$ we can conclude in the following form for the Jaumman rate of Cauchy stress:

$$\overset{\nabla}{\boldsymbol{\sigma}} = \mathbf{R}(t) \cdot \left[\hat{\mathbf{C}}(t) : \left(\mathbf{R}^T(t) \cdot \mathbf{D}(t) \cdot \mathbf{R}(t)\right) \cdot \mathbf{R}^T(t)\right] = \mathbf{C}(t) : \mathbf{D}(t) = \mathbf{C}(t) : \mathbf{L}(t) \tag{4.33}$$

where $\mathbf{C}$ and $\hat{\mathbf{C}}$ are related by:

$$C_{ijkl} = R_{im} R_{jn} R_{kp} R_{lq} \hat{C}_{mnpq}$$

Furthermore, we can express $\boldsymbol{\sigma}$ alternatively as:

$$\begin{aligned} \dot{\boldsymbol{\sigma}} &= \overset{\nabla}{\boldsymbol{\sigma}} - \frac{1}{2}\boldsymbol{\sigma} \cdot (\mathbf{L} - \mathbf{L}^T) + \frac{1}{2}(\mathbf{L} - \mathbf{L}^T) \cdot \boldsymbol{\sigma} \Rightarrow \\ \dot{\boldsymbol{\sigma}} &= \mathbf{C} : \mathbf{L} - \frac{1}{2}\boldsymbol{\sigma} \cdot (\mathbf{L} - \mathbf{L}^T) + \frac{1}{2}(\mathbf{L} - \mathbf{L}^T) \cdot \boldsymbol{\sigma} \end{aligned} \tag{4.34}$$

The last expression in equation (4.34) can be used to approximate $d\boldsymbol{\sigma}$ as:

$$\boxed{d\boldsymbol{\sigma} \cong \mathbf{C} : d\mathbf{L} - \frac{1}{2}\boldsymbol{\sigma} \cdot (d\mathbf{L} - d\mathbf{L}^T) + \frac{1}{2}(d\mathbf{L} - d\mathbf{L}^T) \cdot \boldsymbol{\sigma}} \tag{4.35}$$

Substituting the above approximation for $d\boldsymbol{\sigma}$ into (4.36) we take the following expression for dG:

$$dG = \int_{V(t)} \delta\mathbf{L}^* : \left[\mathbf{C} : d\mathbf{L} - \frac{1}{2}\boldsymbol{\sigma}\cdot(d\mathbf{L} - d\mathbf{L}^T) + \frac{1}{2}(d\mathbf{L} - d\mathbf{L}^T)\cdot\boldsymbol{\sigma} + dL_{kk}\boldsymbol{\sigma}\right] dV$$

or equivalently:

$$\boxed{dG = \int_{V(t)} \delta\mathbf{L}^* : [\mathbf{C} + \boldsymbol{\Sigma} + \boldsymbol{\sigma\delta}] : d\mathbf{L}dV} \tag{4.36}$$

where: $\Sigma_{ijkl} = \frac{1}{2}(\delta_{ik}\sigma_{jl} - \delta_{il}\sigma_{jk} - \delta_{ik}\sigma_{jl} + \delta_{il}\sigma_{jk})$

Now we return back to the finite element approximation summarized by the expressions (4.12)–(4.13). Let us also approximate $\delta\mathbf{L}^*$ and $d\mathbf{L}$ as:

$$\underset{6\times 1}{\{\delta L^*\}} = \underset{6\times n}{[B_L(\mathbf{x})]} \underset{n\times 1}{\left\{\Delta v^{*N}\right\}} \tag{4.37}$$

$$\underset{6\times 1}{\{dL\}} = \underset{6\times n}{[B_L(\mathbf{x})]} \underset{n\times 1}{\left\{\Delta v^{N}\right\}} \tag{4.38}$$

Now if we substitute (4.12), (4.13), (4.37) and (4.38) to (4.36) we will eventually derive:

$$dG = [\Delta v^{*N}]\left[\mathop{\mathrm{A}}_{e=1}\int_{V_e} [B_L]^T\left([\mathbf{C}] + [\boldsymbol{\Sigma}] + \{\sigma\}[\delta]\right)[B_L]dV^e\right]\{\Delta v^N\} \tag{4.39}$$

Note that since the global Jacobian is made up by assembling the Jacobian is defined within each element as:

$$[K] = \mathop{\mathrm{A}}_{e=1}[k^e], \tag{4.40}$$

we can express the Local Jacobian for every element as:

$$\boxed{[k^e] = \int_{V_e} [B_L]^T([\mathbf{C}] + [\boldsymbol{\Sigma}] + \{\sigma\}[\delta])[B_L]dV^e} \tag{4.41}$$

We should also note that the above expression for the Local Jacobian is approximate not only because we introduced the Finite Element approximation for the quantities involved, but also because of the approximation for $d\boldsymbol{\sigma}$ expressed in (4.35). However, this approximation

affects only the rate of convergence and not the solution. Moreover, we should note that matrix $[\boldsymbol{\Sigma}]$ is symmetric and the product $\{\sigma\}[\delta]$ defines a non-symmetric matrix. Therefore, the Local Jacobian $[k^e]$ is considered to be non-symmetric as well.

Abaqus/Standard is based on the formulation presented above to make calculations. Also, ABAQUS enables users to introduce nonlinear elements (UEL) and more complex constitutive material behaviors (UMAT) though subroutines written in FORTRAN form. We will discuss about those user-subroutines later on this chapter.

We will now proceed by representing two types of 2^{nd} order Lagrangian finite elements.

4.4 9-node (2D), 2^{nd} order Lagrangian finite element

Let us consider a 9-node 2D Lagrangian finite element as shown in the figure 4.1.

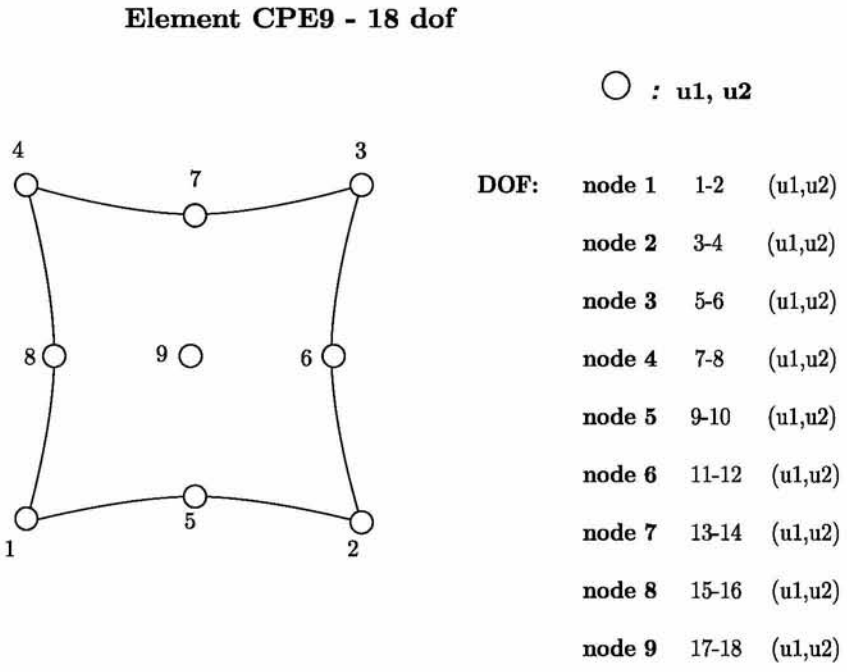


Fig. 4.1: Illustration of a 9-node Lagrangian finite element along with the dof on each node.

Before we apply the Finite Element formulation, we need to recall the **Weak Formulation** in equation (4.9) so that we can determine which quantities need further simplification.

$$G(\boldsymbol{v}) = \int_{V(t)} \boldsymbol{\sigma} : \boldsymbol{D}^* dV - \int_{S(t)} \hat{\boldsymbol{t}} \cdot \boldsymbol{v}^* dS - \int_{V(t)} \rho \boldsymbol{b} \cdot \boldsymbol{v}^* dV = 0 \tag{4.42}$$

We begin with the notations used concerning the Cauchy stress tensor, the traction forces and the body forces. In the case of plain strain those quantities are considered as:

$$\boldsymbol{\sigma} \rightarrow \underset{4\times1}{\{\sigma\}} \qquad \hat{\boldsymbol{t}} \rightarrow \underset{2\times1}{\{t\}} \qquad \boldsymbol{b} \rightarrow \underset{2\times1}{\{b\}} \tag{4.43}$$

Now, we will proceed in the interpolation of the unknowns. The main unknowns of the problem are the nodal displacement which are equal to 18 (9 nodes $\times$ 2 displacements/node) for such an element.

$$\underset{2\times1}{\{\boldsymbol{v}^*(\mathbf{x})\}} = \underset{2\times n}{[N(\mathbf{x})]}\,\underset{n\times1}{\{d^e\}}, \qquad \text{where:} \qquad \underset{1\times2}{\{d^e\}} = [u_1^{[i]} \quad u_2^{[i]}], \qquad i = 1 \rightarrow 9$$

Quantities such as the matrix $[B(\mathbf{x})]$, which contains the spatial derivatives of the shape functions, $N(\mathbf{x})$, and the matrix [N($\mathbf{x}$)], which is the interpolation matrix that consists of the user-defined 'shape' functions, need to be determined. Therefore, for a 9-node Lagrangian finite element:

- The $[B(\mathbf{x})]$ matrix will be of the following form:

$$\underset{4\times n}{[B(\mathbf{x})]} = \begin{bmatrix} \underset{4\times2}{[B_1]} & \underset{4\times2}{[B_2]} & \underset{4\times2}{[B_3]} & \underset{4\times2}{[B_4]} & \underset{4\times3}{[B_5]} & \underset{4\times3}{[B_6]} & \underset{4\times3}{[B_7]} & \underset{4\times3}{[B_8]} & \underset{4\times3}{[B_9]} \end{bmatrix}$$

$$\text{where:} \qquad \underset{4\times2}{[B_i]} = \begin{bmatrix} \dfrac{\partial N_i}{\partial x} & 0 \\ 0 & \dfrac{\partial N_i}{\partial y} \\ 0 & 0 \\ \dfrac{\partial N_i}{\partial y} & \dfrac{\partial N_i}{\partial x} \end{bmatrix}, \qquad i = 1 \rightarrow 9$$

- whereas the $[N(\mathbf{x})]$ matrix will be of the following form:

$$\underset{2\times n}{[N(\mathbf{x})]} = \begin{bmatrix} \underset{2\times2}{[N_1]} & \underset{2\times2}{[N_2]} & \underset{2\times2}{[N_3]} & \underset{2\times2}{[N_4]} & \underset{2\times2}{[N_5]} & \underset{2\times2}{[N_6]} & \underset{2\times2}{[N_7]} & \underset{2\times2}{[N_8]} & \underset{2\times2}{[N_9]} \end{bmatrix}$$

$$\text{where} \qquad \underset{2\times2}{[N_i]} = \begin{bmatrix} N_i & 0 \\ 0 & N_i \end{bmatrix}, \qquad i = 1 \rightarrow 9$$

As far as the user-defined 'shape' functions concerned, for a 9-node Lagrangian finite element [2] they are defined as:

[2] For a more analytical description about the form of the shape-functions in a 9-node Lagrangian Finite Element you Zienkiewicz[30]

$$
\begin{aligned}
N_1 &= \frac{1}{4}\cdot(\xi^2-\xi)\cdot(\eta^2-\eta) \quad N_2 = \frac{1}{4}\cdot(\xi^2+\xi)\cdot(\eta^2-\eta)\\
N_3 &= \frac{1}{4}\cdot(\xi^2+\xi)\cdot(\eta^2+\eta) \quad N_4 = \frac{1}{4}\cdot(\xi^2-\xi)\cdot(\eta^2+\eta)\\
N_5 &= \frac{1}{2}\cdot(1-\xi^2)\cdot(\eta^2-\eta) \quad N_6 = \frac{1}{2}\cdot(\xi^2+\xi)\cdot(1-\eta^2)\\
N_7 &= \frac{1}{2}\cdot(1-\xi^2)\cdot(\eta^2+\eta) \quad N_8 = \frac{1}{2}\cdot(\xi^2-\xi)\cdot(1-\eta^2)\\
N_9 &= (1-\xi^2)\cdot(1-\eta^2)
\end{aligned}
\tag{4.44}
$$

where (ξ, η) is a local coordinate system[30].

We conclude that the **Weak Formulation** (4.9) and the equation which gives us the **Jacobian** (4.41) of an element, remain the same in the case of a 9-node Lagrangian finite element with plain strain. There are only little changes regarding the dimensions of the matrices that take place on the integrals.

4.5 9-node (2D) 2^{nd} order 'Mixed' Lagrangian finite element

Now let us discuss the implementation of the finite element analysis in the case of a 9-node 2D, 2^{nd} order 'Mixed' Lagrangian finite element. The term 'mixed' denotes the use of a mixture of displacement and stress variables with an augmented variational principle to approximate the equilibrium equations and compatibility conditions. The main unknowns of the problem are the displacement variables and the pressure. A schematic representation of a 9-node 'mixed' Lagrangian finite element is shown in figure 4.2

This type of element is used when the material response is incompressible and the solution of the problem cannot be obtained in terms of the displacement history only, since a purely hydrostatic pressure can be added without changing the displacements. An example where this behaviour approached in a system, is the nearly incompressible case in which Poisson's ratio is greater than 0.4999999. On a nearly incompressible case, a very small change in displacement produces extremely large changes in pressure, so that a displacement-based solution is too sensitive numerically. Therefore, in order to avoid this singular behavior in the system we treat the pressure stress as an independently interpolated basic solution variable, coupled to the displacement solution through the constitutive theory and the compatibility conditions.

When a 'mixed' finite element is used in an analysis, we use the deviatoric part of the Cauchy stress (Zienkiewicz[30]), $\boldsymbol{s}$, in the problem. Recall that the deviatoric part of $\boldsymbol{\sigma}$ is given by equation $\boldsymbol{s} = \boldsymbol{\sigma} - p\boldsymbol{\delta}$. Due to the existence of the pressure in the "mixed formulation" model, we need to alter the general form of the **Weak Formulation** and the **Jacobian**. To begin with, we write again the **Weak Formulation** taking into account the existence of pressure:

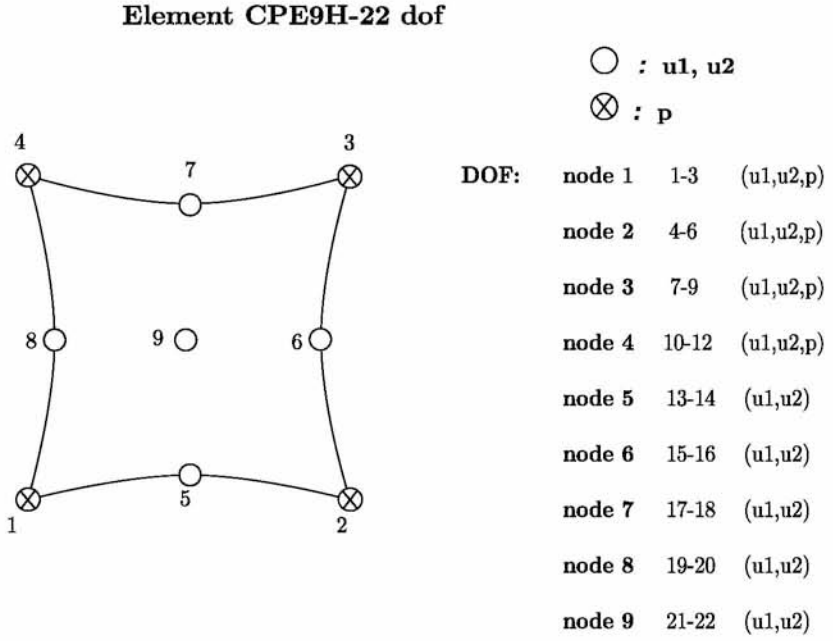


Fig. 4.2: Illustration of a 9-node "mixed" Lagrangian finite element along with the dof on each node.

$$\int_{V(t)} \underbrace{\left(\sigma_{ij,j} + \rho b_i\right) v_i^*}_{(I)} dV + \int_{V(t)} \left(\frac{p}{\kappa} - u_{i,i}\right) p^* dV + \int_{S(t)} \left(\hat{t}_i - \sigma_{ij} n_j\right) v_i^* dS = 0 \tag{4.45}$$

The quantity (I) in the first integral is calculated as follows:

$$(I) = \int_{V(t)} \sigma_{ij,j} v_i^* dV = \int_{V(t)} \left[\left(\sigma_{ij} v_i^*\right)_{,j} - \sigma_{ij} v_{i,i}^*\right] dV = \int_{S(t)} \sigma_{ij} v_i^* n_j dS - \int_{V(t)} \sigma_{ij} v_i^* dV \tag{4.46}$$

Substituting equation (4.46) into (4.45) we conclude in the final form of the **Weak Formulation**:

$$\int_{V(t)} \left[s_{ij} D_i^* + p v_{ij}^*\right] dV + \int_{V(t)} \left(u_{i,i} - \frac{p}{\kappa}\right) p^* dV - \int_{V(t)} \rho b_i v_i^* dV - \int_{S(t)} \hat{t}_i v_i^* dS = 0 \tag{4.47}$$

The interpolation of the unknowns according to the finite element analysis is as follows:

1. The arbitrary velocity field is given by

$$\underset{2\times 1}{\{\delta v^*\}} = \underset{2\times n}{[N_v(\mathbf{x})]} \underset{n\times 1}{\{d^{e*}\}} \tag{4.48}$$

where $[N_v(\mathbf{x})]$ is the interpolation matrix consisting of the user-defined 'shape' functions, N_i:

$$\underset{2\times n}{[N_v(\mathbf{x})]} = \begin{bmatrix} \underset{2\times 3}{[N_{v1}]} & \underset{2\times 3}{[N_{v2}]} & \underset{2\times 3}{[N_{v3}]} & \underset{2\times 3}{[N_{v4}]} & \underset{2\times 2}{[N_{v5}]} & \underset{2\times 2}{[N_{v6}]} & \underset{2\times 2}{[N_{v7}]} & \underset{2\times 2}{[N_{v8}]} & \underset{2\times 2}{[N_{v9}]} \end{bmatrix} \tag{4.49}$$

where

$$\underset{2\times 3}{[N_{vi}]} = \begin{bmatrix} N_i & 0 & 0 \\ 0 & N_i & 0 \end{bmatrix}, \quad i = 1 \to 4 \quad \text{and} \quad \underset{2\times 2}{[N_{vi}]} = \begin{bmatrix} N_i & 0 \\ 0 & N_i \end{bmatrix}, \quad i = 5 \to 9$$

The shape functions, N_i, have already declared in the previous section, in equation (4.44)

2. The pressure is interpolated as

$$\underset{1\times 1}{p^*(\mathbf{x})} = \underset{1\times n}{[N_p(\mathbf{x})]} \underset{n\times 1}{\{d^{e*}\}}$$

where $[N_p(\mathbf{x})]$ is the interpolation matrix of the user-defined 'shape' functions, $\hat{N}_i$. The shape functions $\hat{N}_i$, are given by the following equations:

$$\hat{N}_1 = \frac{1}{4}(1-\xi)(1-\eta) \qquad \hat{N}_2 = \frac{1}{4}(1+\xi)(1-\eta)$$
$$\hat{N}_3 = \frac{1}{4}(1+\xi)(1+\eta) \qquad \hat{N}_4 = \frac{1}{4}(1-\xi)(1+\eta)$$

3. The interpolation of the virtual velocity field, D^*, is as follows:

$$\underset{4\times 1}{\{\delta D^*\}} = \underset{4\times n}{[B_{dev}(\mathbf{x})]} \underset{n\times 1}{\{d^{e*}\}}$$

where $[B_{dev}(\mathbf{x})]$ is given as follows:

$$\underset{4\times n}{[B_{dev}(\mathbf{x})]} = \begin{bmatrix} \underset{4\times 4}{[B_1]} & \underset{4\times 4}{[B_2]} & \underset{4\times 4}{[B_3]} & \underset{4\times 4}{[B_4]} & \underset{4\times 3}{[B_5]} & \underset{4\times 3}{[B_6]} & \underset{4\times 3}{[B_7]} & \underset{4\times 3}{[B_8]} & \underset{4\times 3}{[B_9]} \end{bmatrix}$$

where $[B_i]$ is in the following form:

$$[B_i]_{4\times3} = \begin{bmatrix} \frac{2}{3}\frac{\partial N_i}{\partial x} & -\frac{1}{3}\frac{\partial N_i}{\partial y} & 0 \\ -\frac{1}{3}\frac{\partial N_i}{\partial x} & \frac{2}{3}\frac{\partial N_i}{\partial y} & 0 \\ -\frac{1}{3}\frac{\partial N_i}{\partial x} & -\frac{1}{3}\frac{\partial N_i}{\partial y} & 0 \\ \frac{\partial N_i}{\partial y} & \frac{\partial N_i}{\partial x} & 0 \end{bmatrix} \quad i = 1 \rightarrow 4 \qquad [B_i]_{4\times2} = \begin{bmatrix} \frac{2}{3}\frac{\partial N_i}{\partial x} & -\frac{1}{3}\frac{\partial N_i}{\partial y} \\ -\frac{1}{3}\frac{\partial N_i}{\partial x} & \frac{2}{3}\frac{\partial N_i}{\partial y} \\ -\frac{1}{3}\frac{\partial N_i}{\partial x} & -\frac{1}{3}\frac{\partial N_i}{\partial y} \\ \frac{\partial N_i}{\partial y} & \frac{\partial N_i}{\partial x} \end{bmatrix} \quad i = 5 \rightarrow 9$$

4. We continue with the interpolation of the quantity $v^*_{i,i}(\mathbf{x})$

$$\underset{1\times1}{v^*_{i,i}} = \underset{1\times n}{[B_v(\mathbf{x})]}\,\underset{n\times1}{\{d^e\}}$$

where $[B_v(\mathbf{x})]$ is a matrix consisting of the derivatives of the shape functions N_i and it can be written as follows:

$$\underset{1\times n}{[B_v(\mathbf{x})]} = \begin{bmatrix} \underset{1\times3}{[B_{v1}]} & \underset{1\times3}{[B_{v2}]} & \underset{1\times3}{[B_{v3}]} & \underset{1\times3}{[B_{v4}]} & \underset{1\times3}{[B_{v5}]} & \underset{1\times3}{[B_{v6}]} & \underset{1\times3}{[B_{v7}]} & \underset{1\times3}{[B_{v8}]} & \underset{1\times3}{[B_{v9}]} \end{bmatrix}$$

and the matrices $[B_{vi}]$ are in the following form:

$$\underset{1\times3}{[B_{vi}]} = \begin{bmatrix} \frac{\partial N_i}{\partial x} & \frac{\partial N_i}{\partial y} & 0 \end{bmatrix} \quad i = 1 \rightarrow 4 \qquad \underset{1\times2}{[B_{vi}]} = \begin{bmatrix} \frac{\partial N_i}{\partial x} & \frac{\partial N_i}{\partial y} \end{bmatrix} \quad i = 5 \rightarrow 9$$

where $[\boldsymbol{s}]_{1\times4} = [s_{11} \;\; s_{22} \;\; s_{33} \;\; \sigma_{12}]$ is determined in terms of the $\boldsymbol{F}_{n+1}$ in a User Subroutine (**UMAT**) which is described in subsection 4.7.

4.6 Description of UEL subroutine

User defined **EL**ement subroutines (UEL) are used in conjunction with Abaqus [2] allowing the users to define any (nonlinear) element of arbitrary complexity. For this reason, ABAQUS has a DO loop over all elements. Within this loop, **UEL** is called for each element and it is expected to provide the element residual ("load vector") and the element Jacobian ("stiffness matrix"). At the end of this loop, the global residual load vector $\{f\}_{n+1}$ and the global Jacobian $[k] = -\frac{\partial\{f\}}{\partial d}$ have been formed. If the residual is not within the specified tolerance, a correction to the nodal degrees of freedom (dof) is calculated and the loop over all elements is repeated, until the global residual is within tolerance (global 'equilibrium' convergence).

Within **UEL** we have a DO loop over the element Gauss integration points where the element 'load vector' $\{f^e\}_{n+1}$ and the element Jacobian $[k^e] = -\frac{\partial f^e}{\partial d^e}$ are calculated as they will be used to estimate the element nodal dof $\{d^e\}_{n+1}$. Then $\{d^e\}_{n+1}$ for the current increment is passed in together with the values of the element nodal dof at the end of the previous increment $\{d^e\}_n$.

Note that in order to form the element Jacobian we need to calculate $\boldsymbol{\sigma}_{n+1}, \Delta\bar{\epsilon}^p_{n+1}$ and the derivatives $\partial\boldsymbol{\sigma}_{n+1}/\partial\varepsilon_{n+1}$. For this calculation, we call another subroutine named **UMAT** which we present later on in this chapter.

Defining a User Element

There is a large number of properties that need to be defined before we begin coding. Those properties are:

- The number of nodes on the element
- The number of coordinates present at each element
- The degrees of freedom active at each node
- The number of element properties to be defined external to the **UEL**
- The number of solution-dependent state variables (SDVs) to be stored per element
- The number of (distributed) load types available for the element

The main contribution of the **U**ser defined **EL**ements, when the degrees of freedom of the problem are the displacements of the nodes, u^N, is to provide the nodal residual forces F^N. The nodal residual forces F^N are defined through equation $F^N = F^N_{ext} - F^N_{int}$, where F^N_{ext} is the external load due to applied distributed loads and F^N_{int} is the internal load due to stresses, e.g., at node N.

In nonlinear user elements the forces will often depend on the increments in the degrees of freedom Δu^N and the internal state variables which are denoted as H^α. In order to obtain the solution of the (nonlinear) system of equations it is required to define the element Jacobian (stiffness matrix):

$$K^{NM} = -\frac{dF^N}{du^M} \tag{4.50}$$

The Jacobian should include all the direct and indirect dependencies of F^N on u^N, which includes terms of the form:

$$-\frac{\partial F^N}{\partial H^\alpha}\frac{\partial H^\alpha}{\partial u^M}$$

We have to note that the Jacobian can be either symmetric or nonsymmetric. An accurately defined Jacobian will always improve convergence with the associated computational cost.

A typical interface for a **UEL** subroutine is given below:

```
1
2       SUBROUTINE UEL(RHS,AMATRX,SVARS,ENERGY,NDOFEL,NRHS,NSVARS,
3      + PROPS,NPROPS,COORDS,MCRD,NNODE,U,DU,V,A,JTYPE,TIME,DTIME,
4      + KSTEP,KINC,JELEM,PARAMS,NDLOAD,JDLTYP,ADLMAG,PREDEF,NPREDF,
5      + LFLAGS,MLVARX,DDLMAG,MDLOAD,PNEWDT,JPROPS,NJPROP,PERIOD)
6 C
7         INCLUDE 'ABA_PARAM.INC'
8 C
9         DIMENSION RHS(MLVARX,*),AMATRX(NDOFEL,NDOFEL),PROPS(*),
10       + SVARS(*),ENERGY(8),COORDS(MCRD,NNODE),U(NDOFEL),
11       + DU(MLVARX,*),V(NDOFEL),A(NDOFEL),TIME(2),PARAMS(*),
12       + JDLTYP(MDLOAD,*),ADLMAG(MDLOAD,*),DDLMAG(MDLOAD,*),
13       + PREDEF(2,NPREDF,NNODE),LFLAGS(*),JPROPS(*)
14
15
16        user coding to define RHS, AMATRX, SVARS, ENERGY, and PNEWDT
17
18
19        RETURN
20        END
```

4.7 Description of a UMAT subroutine

User **MAT**erial subroutines (UMAT) allow users to implement general constitutive equations of non-linear materials in a programming language such as FORTRAN. **UMAT** can also be used within a **UEL** where it is called for every integration point within each element.

Recalling the FEM approximation that we introduced in Section 4.2, the unknown displacement field is approximated by a function interpolation defined within each element. The main unknown of the problem are the nodal displacements. Stresses and strains are being calculated at the integration point of each element.[3] In a UMAT subroutine, the user has to provide the complete set of calculations that are necessary to incrementally update the stresses and also provide the consistent tangent $(d\boldsymbol{\sigma}/d\boldsymbol{\varepsilon})$ both of which depend on the constitutive model under consideration as well as the integration algorithm used for numerical integration. Concerning the solution dependent state variables (STATEV), they can be variables necessary to conduct material calculations at a given time or even variable whose values are needed to be stored for post processing usage. ABAQUS updates and stores state variables at the end of every increment for future calculations.

Below we represent a typical interface of a UMAT subroutine.

```
1         SUBROUTINE UMAT(STRESS,STATEV,DDSDDE,SSE,SPD,SCD,
```

[3] Integration points and their position within the element both depend on the type of element being used.

```
      + RPL,DDSDDT,DRPLDE,DRPLDT,
      + STRAN,DSTRAN,TIME,DTIME,TEMP,DTEMP,PREDEF,DPRED,CMNAME,
      + NDI,NSHR,NTENS,NSTATV,PROPS,NPROPS,COORDS,DROT,PNEWDT,
      + CELENT,DFGRD0,DFGRD1,NOEL,NPT,LAYER,KSPT,KSTEP,KINC)
C
      INCLUDE 'ABA_PARAM.INC'
C
      CHARACTER*80 CMNAME
      DIMENSION STRESS(NTENS),STATEV(NSTATV),
      + DDSDDE(NTENS,NTENS),DDSDDT(NTENS),DRPLDE(NTENS),
      + STRAN(NTENS),DSTRAN(NTENS),TIME(2),PREDEF(1),DPRED(1),
      + PROPS(NPROPS),COORDS(3),DROT(3,3),DFGRD0(3,3),DFGRD1(3,3)

      user coding to define DDSDDE, STRESS, STATEV, SSE, SPD, SCD
      and, if necessary, RPL, DDSDDT, DRPLDE, DRPLDT, PNEWDT

      RETURN
      END
```

4.8 A schematic representation on how UEL and UMAT subroutines work

So far we discussed about the finite element approximation and UEL, UMAT subroutines along with two types of elements, the CPE9 and CPE9H. In order to have a better understanding on how a UEL and a UMAT subroutine works and how these two subroutines cooperate, we represent two flowcharts illustrating the procedures to be followed in such subroutines in the case of a 9-node (2D), 2^{nd} order Lagrangian finite element with and without linear pressure in the case of metal plasticity (Figures 4.3 and 4.5).

Figure 4.3 shows how the UEL subroutine works in the case where the degrees of freedom of the model are only the nodal displacements. Through this flowchart we can infer what variables need to be initially determined, where the UMAT subroutine needs to be called and which variables have been calculated at the end of the Do Loop. Figure 4.4 shows how the UMAT subroutine works, which calculations take place in this subroutine and which are the output variables [4].

Figure 4.5 shows how the UEL subroutine works in the case where the degrees of freedom of the model are both the nodal displacements and the pressure [5]. Through this flowchart we can infer what variables need to be initially determined, where the UMAT subroutine needs to be called and which variables have been calculated at the end of the Do Loop. Figure 4.4 shows how the UMAT subroutine works, which calculations take place in this subroutine and which are the output variables.

[4] The calculations and the output variables of UMAT depend on the consitutive model used as well as on the numerical integration scheme

[5] Pressure exists only on the corner nodes of the element. Thus, by including pressure in the model we increase the nodal unknowns by 4

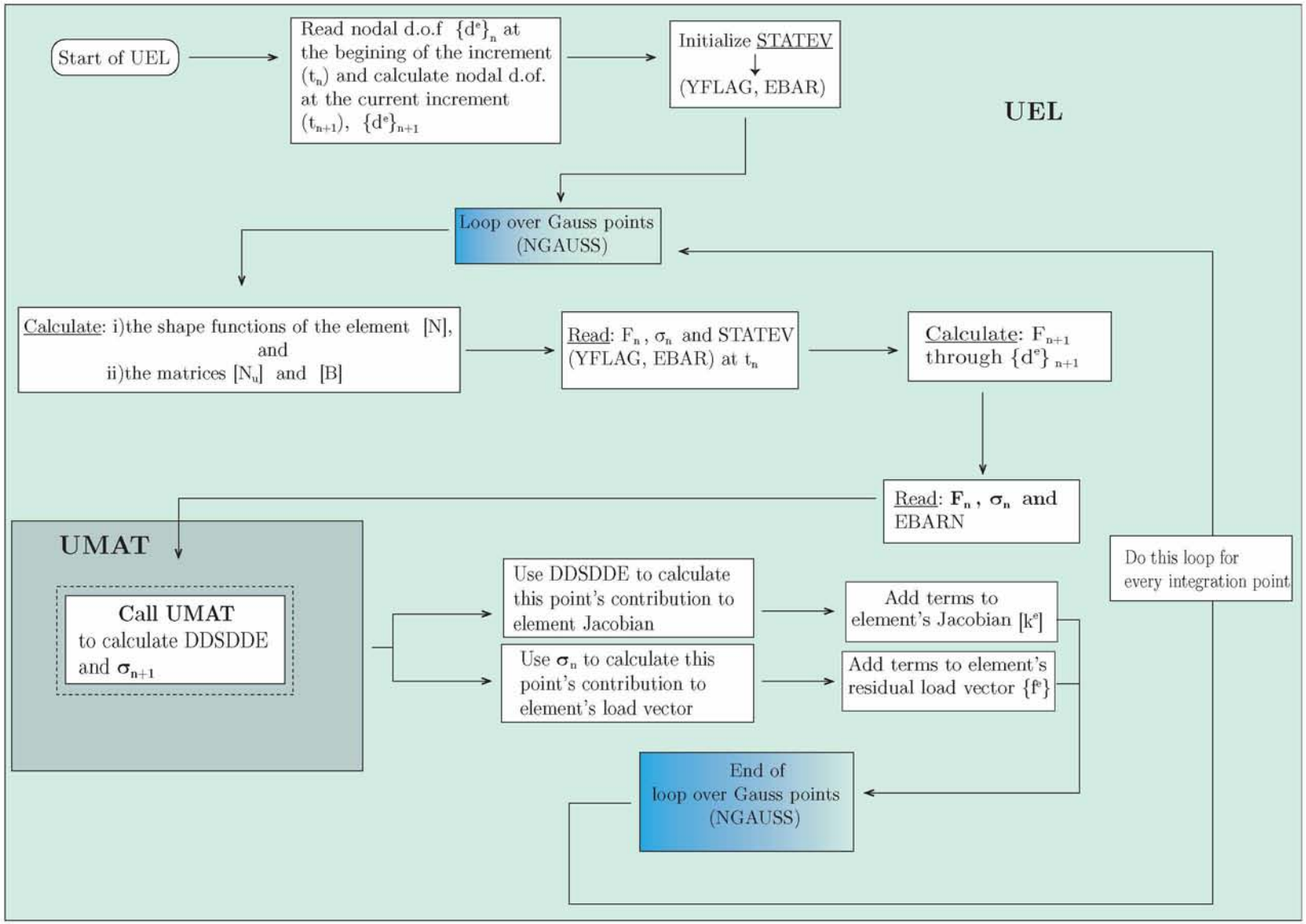


Fig. 4.3: UEL's flowchart for a 9-node Lagrangian Finite Element: A schematic representation on how it works and what variables need to be determined.

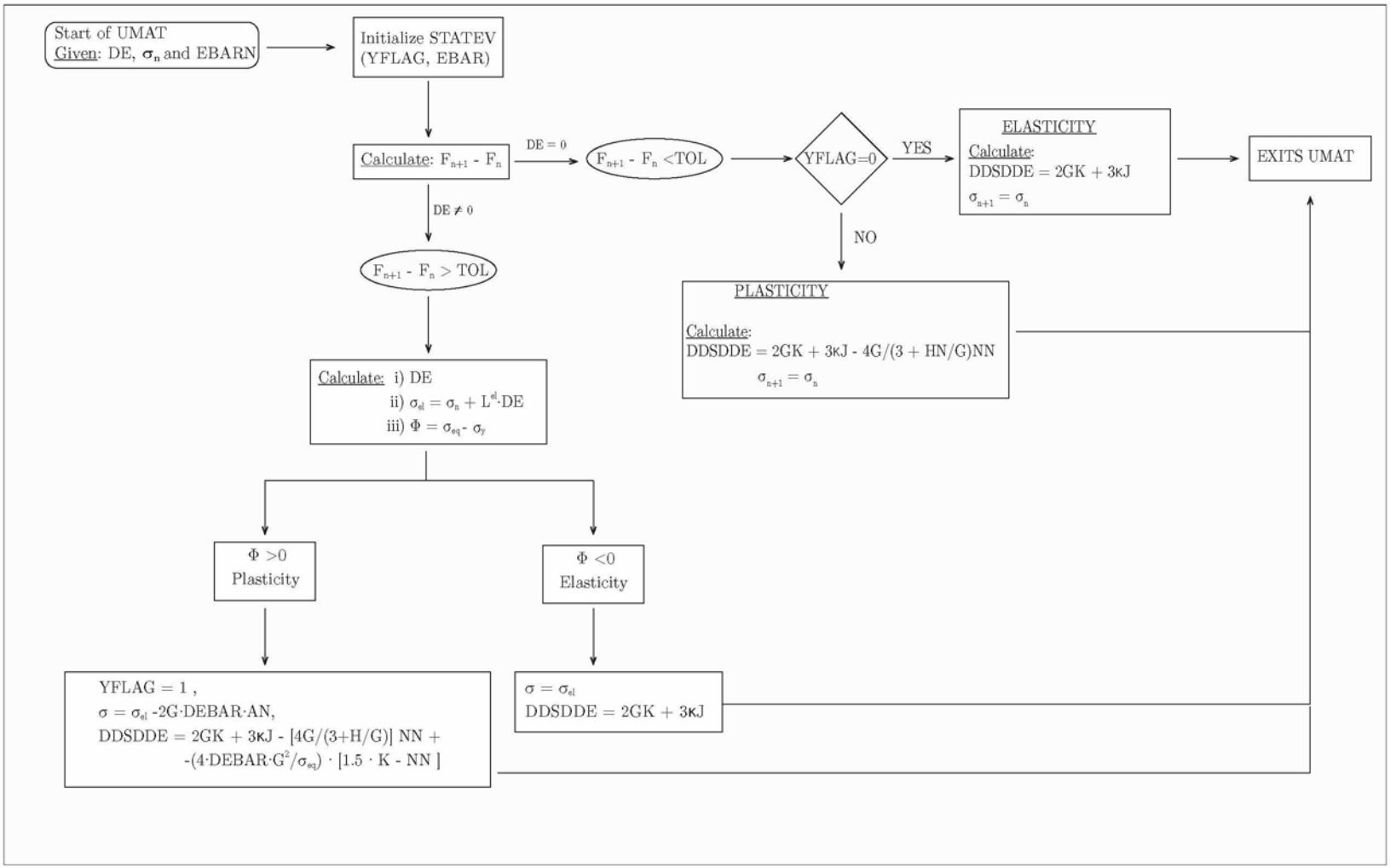


Fig. 4.4: UMAT's flowchart for metal elasticity and plasticity: A schematic representation on how it works and what variables need to be determined.

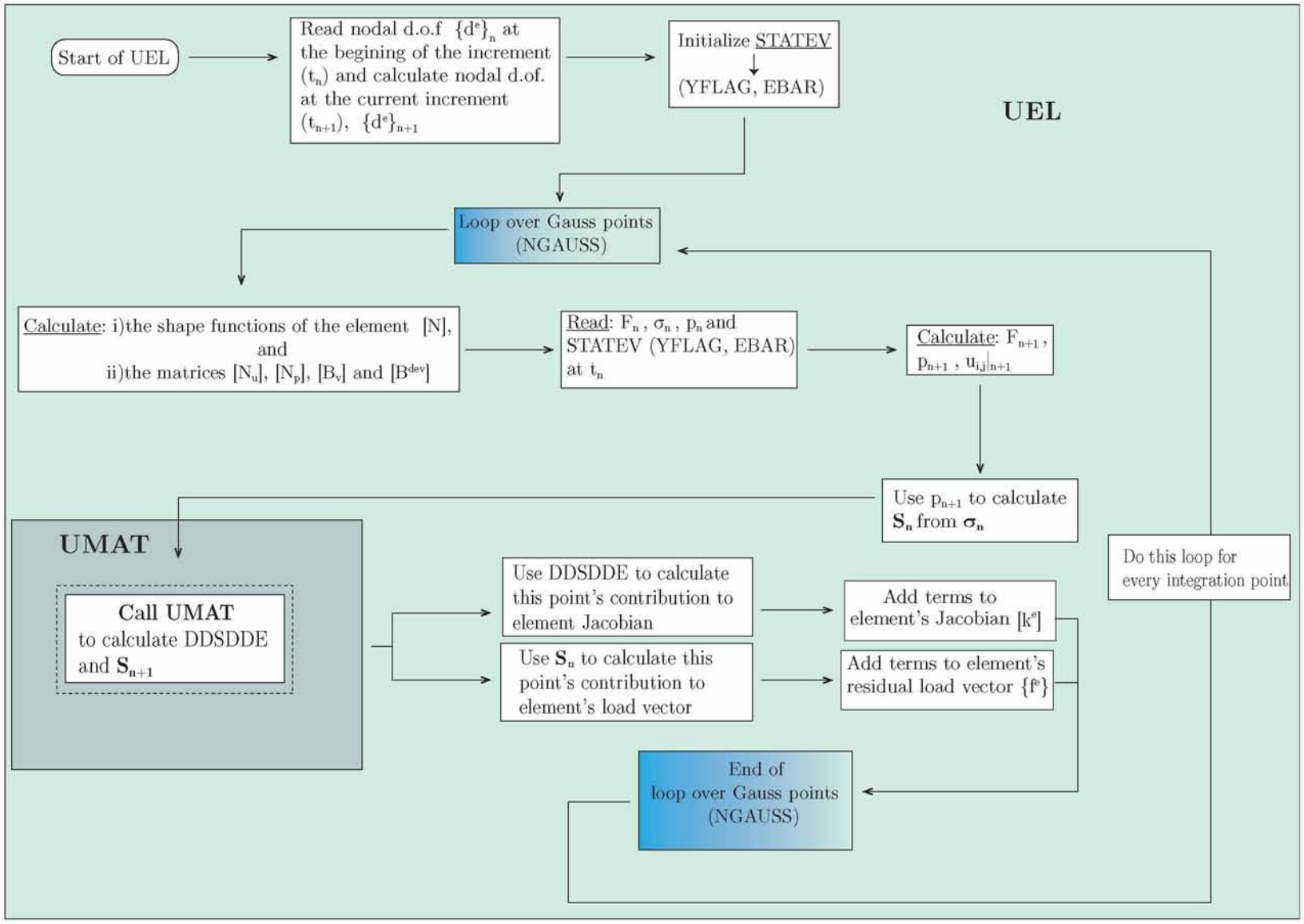


Fig. 4.5: UEL's flowchart for a 9-node Lagrangian Finite mixed Element: A schematic representation on how it works and what variables need to be determined.

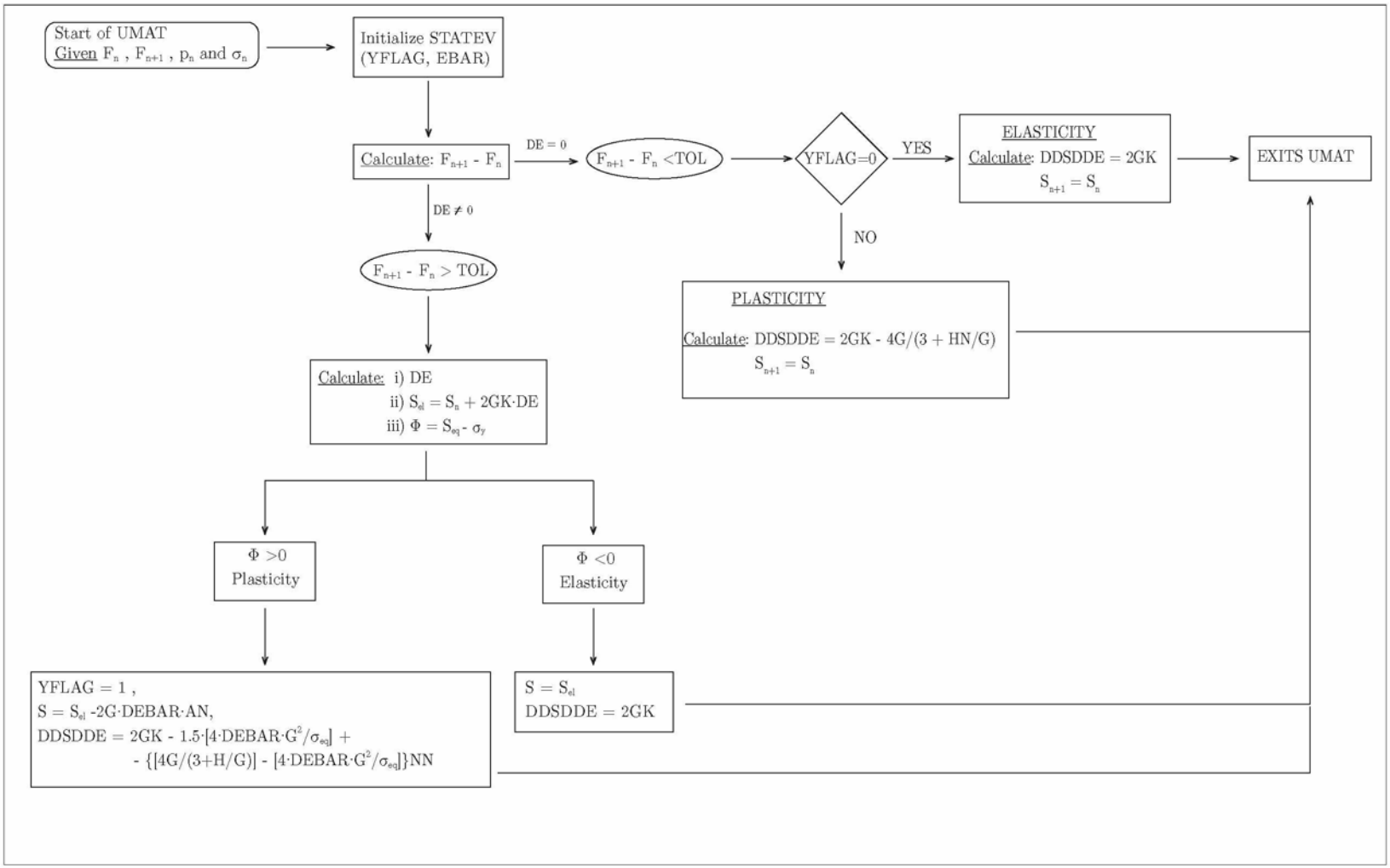


Fig. 4.6: UMAT's flowchart for metal elasticity and plasticity: A schematic representation on how it works and what variables need to be determined.

Chapter 5

Finite Element Results

5.1 Introduction

In the current chapter we will present a variety of finite element simulations using the Abaqus [1] general purpose finite element software in conjunction with the UEL (User Element) subroutine modelling the 9-node Lagrangian finite element that was developed in the context of this thesis. The implementation of finite deformation metal plasticity was possible through the UMAT subroutine which is called for every integration point within one element in UEL[1]. To verify the correctness of the computational implementation of both the simple 9-node Lagrangian element as well as of the 9-node Lagrangian element with the mixed formulation we conducted a series of finite element analyses with increasing complexity as suggested by Abaqus [2]. The finite element simulations discussed in the following section involve a mesh consisting of only 1 finite element and the results are compared with the corresponding ones for a 8-node Serendipity finite element from the Abaqus library.

In the following sections we present and discuss the results from a variety of one element analyses in the cases of uniaxial tension and compression, with force control and displacement control, and in the case of contact between one element and a rigid body.

5.2 One element analysis

In order to validate the computational implementation of user subroutines UEL and UMAT, we conducted a series of simulations using a single finite element in the mesh. We

[1] *see* chapter 4 and section 4.6 for more details regarding these subroutines

tried these single element tests in the case of simple uniaxial tension and compression using both force and displacement control [2]. The reason behind the implementation of force and displacement control in the single finite element tests, is that there is a variety of problems that use either force or displacement control. Therefore, it is reasonable to assert that UEL subroutine works properly in these type of problems.

In our single element analysis, we used 9-node (2D) 2^{nd} order Lagrangian finite elements and 9-node (2D) 2^{nd} order Lagrangian finite elements with the mixed formulation. Moreover, we carried out the same analyses with an 8-node Serendipity finite element from the Abaqus library for comparison purposes. The Abaqus elements that we chose to implement are the *CPE8H* (Continuum Plane–Strain 8–node hybrid finite elements) [2].

For all the analyses performed in this section we assume an elastoplastic metallic material whose properties are listed below,

- Young's modulus,E, was taken to be that of steel : 216 GPa
- Poisson's ratio, ν, was taken to be that of steel : 0.3
- Yield Stress, σ_0, was taken to be that of steel : equals to 850 MPa
- Power law hardening with a hardening exponent of $1/5 = 0.2$

We have to note that for all the one element simulations we normalize E and σ_0 with σ_0. Thus, σ_0 equals to 1 and E equals to 254.118 (E/σ_0).

5.2.i Uniaxial tension with force control and displacement control

During the uniaxial tension with force control, we applied a concentrated force field along the $\mathbf{e}_2$ direction. The undeformed and deformed meshes for the uniaxial tension with force control are shown in figure 5.1

We have to note that the force applied on the middle node on the top surface of the element, is four times larger than the force applied on the two corner nodes. This results from the finite element approximation on 2^{nd} order finite elements[3].

During our analysis we examined three different types of elements: (**i**) a 9-node (2D), 2^{nd} order Lagrangian finite element, (**ii**) a 9-node (2D), 2^{nd} order Lagrangian finite element with linear pressure and (**iii**) the CPE8H (Continuum Plane strain(E) 8-node hybrid finite element) Abaqus element. Next we present the Von Mises stress vs. Logarithmic strain (equivalent) that resulted from each simulation.

[2]Force control implies and a distributed or a concentrated force is applied to the finite element to cause deformation whereas displacement control implies that we prescribe the displacement of a given node set in the form of boundary conditions. In the latter, no force is applied to the finite element

[3]An analytical explanation why the force applied on the middle node is four times bigger than the sided nodes in a 2^{nd} order Lagrangian Finite Element, can be found in Zienkiewicz[30]

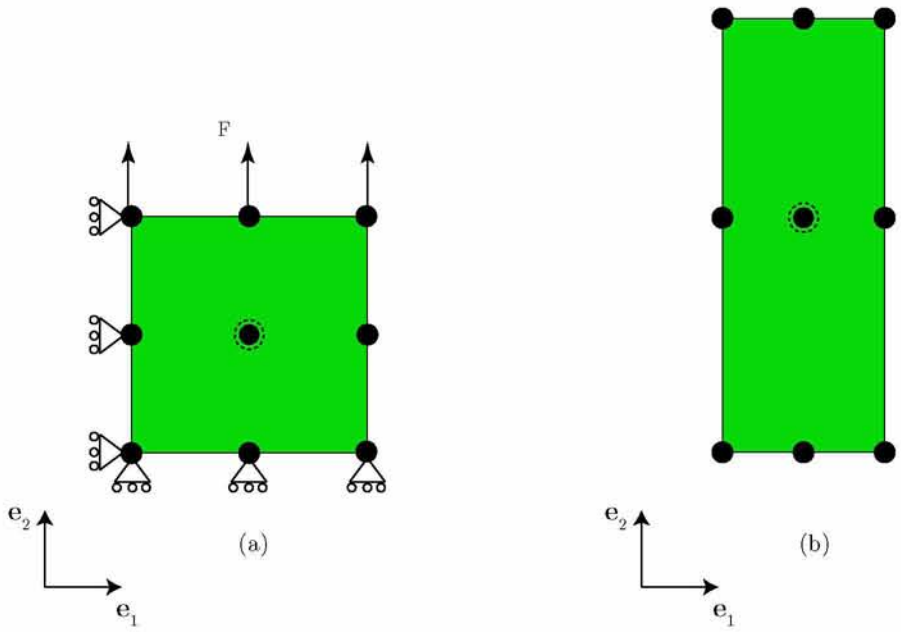


Fig. 5.1: The undeformed (a) and deformed (b) meshes in the case of a single finite element under uniaxial tension with force control. The black points indicate the location of each node in the finite element. The central node only exists in the case of User ELements. The force field is applied on the three topmost nodes of the element and the boundary conditions are such to prevent rigid body motion

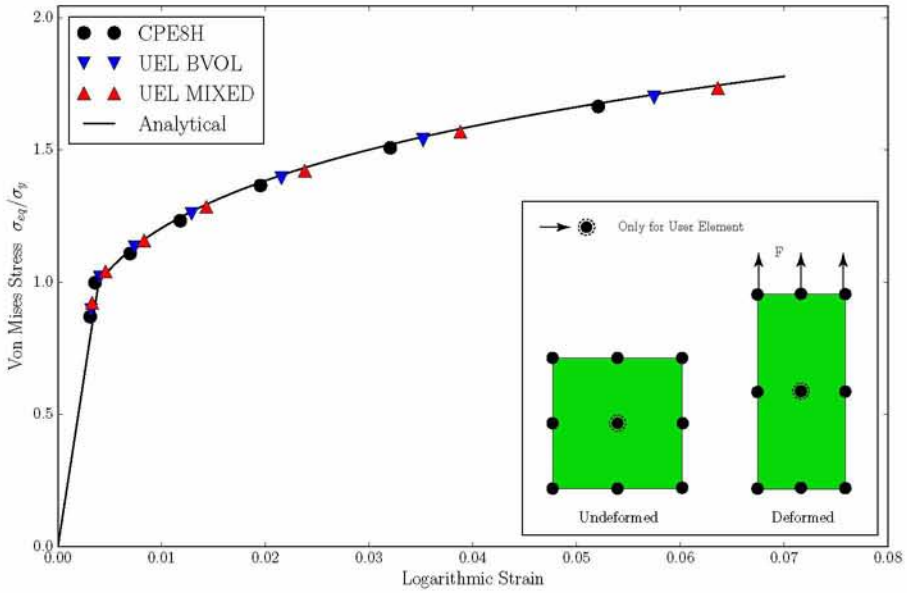


Fig. 5.2: The stress (Von Mises) - strain (equivalent Logarithmic strain) curve during uniaxial tension with force control, using user–defined elements and Abaqus elements along with the curve for the analytical solution in this case

The results shown in figure 5.2 suggest that Abaqus elements and user–defined elements are in agreement with one another as well as with the analytical solution. This verifies the correctness of the computational implementation of both User Elements in the case of

uniaxial tension with force control.

We continue with the case of uniaxial tension with displacement control. In this simulation we applied a uniform displacement field on the upper surface of the element subjecting the element into uniaxial tension. Figure 5.3 shows the undefromed and deformed configurations along with the boundary conditions that used in the analysis. The Von Mises stress – Logarithmic strain (equivalent) curves that resulted from each simulation are reported in Figure 5.4

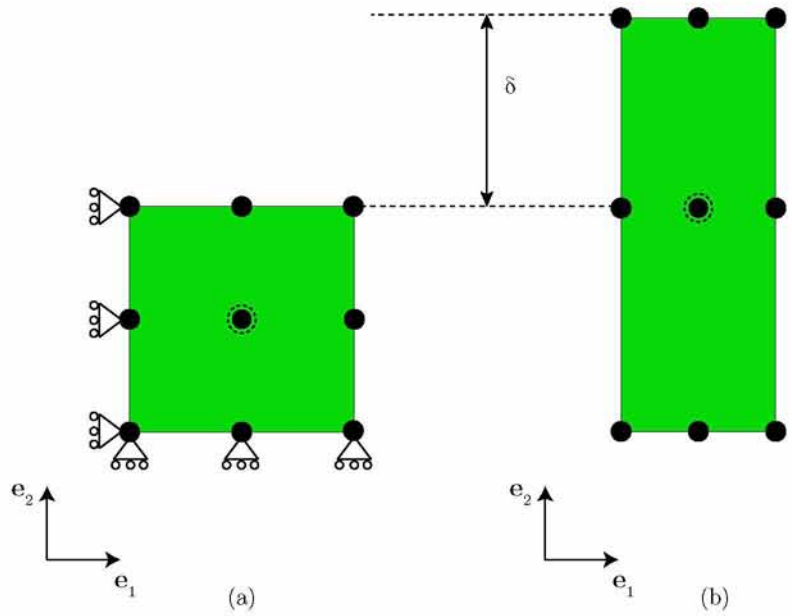


Fig. 5.3: The undeformed (a) and deformed (b) meshes in the case of a single finite element under uniaxial tension with force control. The black points indicate the location of each node in the finite element. The central node only exists in the case of User ELements. The uniform displacement field is applied on the three topmost nodes of the element and the boundary conditions are such to prevent rigid body motion

Figure 5.4 suggests that Abaqus elements and user – defined elements are in good agreement with the analytical solution once again verifying the fact that the user element where implemented correctly in this case as well.

5.2.ii Uniaxial compression with force control and displacement control

During the uniaxial compression with force control, we applied a concentrated compressive force field at the nodes of the upper surface of the element in the direction $\mathbf{e}_2$. The undeformed and deformed configurations are shown in figure 5.5. We used both 9–node user defined elements and Abaqus elements in order to verify our results.

The stress (Von Mises)–strain (equivalent Logarithmic strain) curves that correspond to

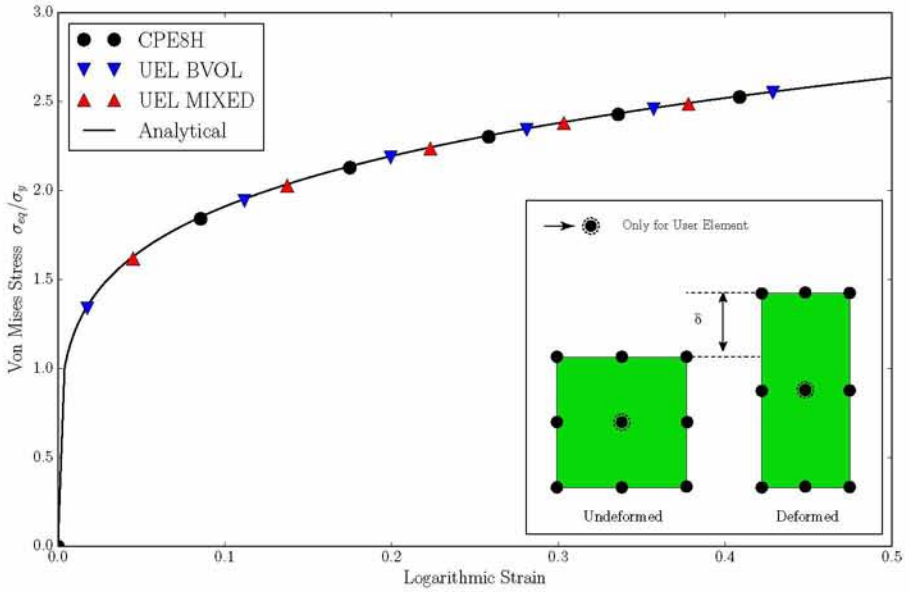


Fig. 5.4: The stress (Von Mises) - strain (equivalent Logarithmic strain) curve during uniaxial tension with displacement control, using user–defined elements and Abaqus elements along with the curve for the analytical solution in this case

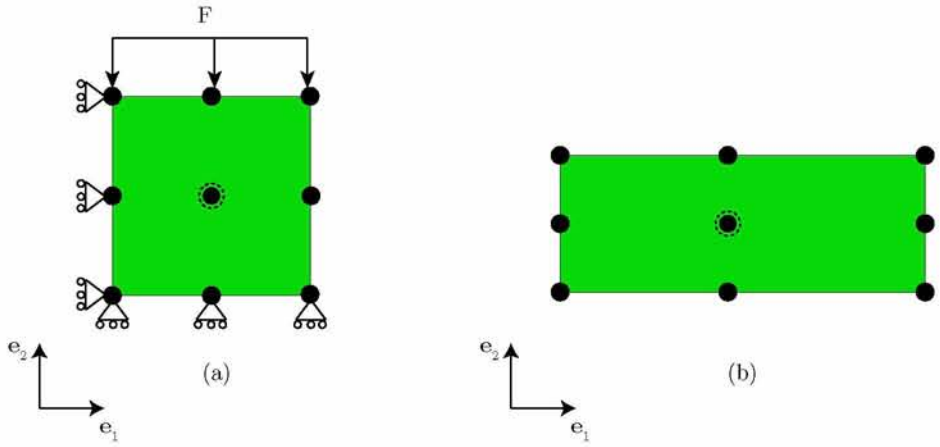


Fig. 5.5: The undeformed (a) and deformed (b) meshes in the case of a single finite element under uniaxial compression with force control. The black points indicate the location of each node in the finite element. The central node only exists in the case of User ELements. The force field is applied on the three topmost nodes of the element and the boundary conditions are such to prevent rigid body motion

each simulation are shown in figure 5.6. From the stress – strain curves we can infer that the results for user – defined elements are in agreement with both the Abaqus Element as well as the analytical solution.

We continue with the case of uniaxial compression with displacement control. Figure 5.7 shows the undeformed and deformed configurations along with the imposed displacement field on the upper surface of the element and the boundary conditions.

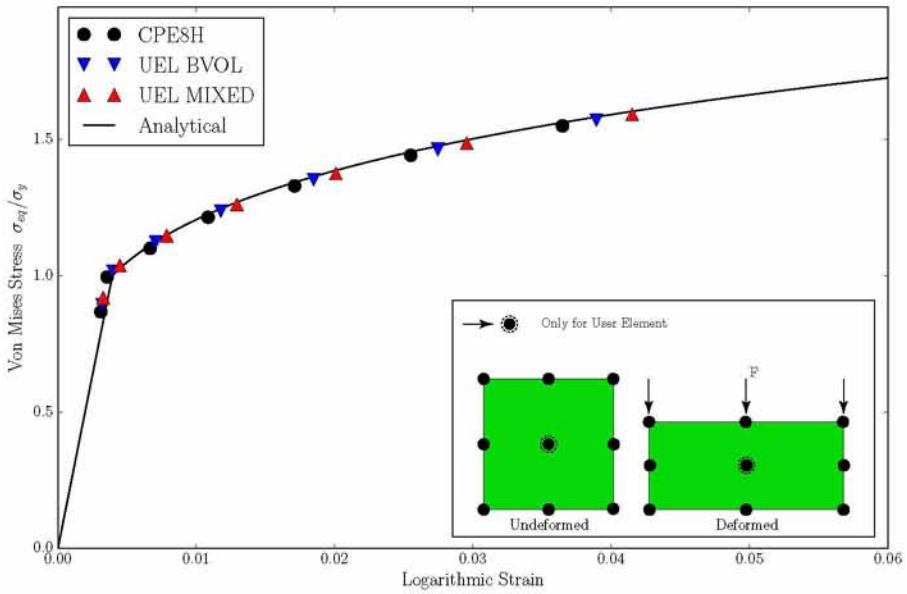


Fig. 5.6: The stress (Von Mises) - strain (equivalent Logarithmic strain) curve during uniaxial tension with displacement control, using user–defined elements and Abaqus elements along with the curve for the analytical solution in this case

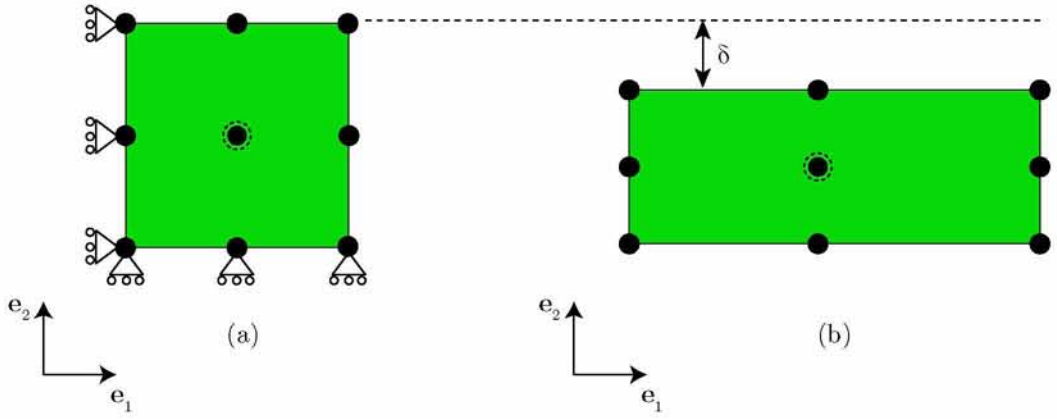


Fig. 5.7: The undeformed (a) and deformed (b) meshes in the case of a single finite element under uniaxial compression with displacement control. The black points indicate the location of each node in the finite element. The central node only exists in the case of User ELements. The displacement field is applied on the three topmost nodes of the element and the boundary conditions are such to prevent rigid body motion

The stress (Von Mises)–strain (equivalent Logarithmic strain) curves that correspond to each simulation are shown in figure 5.8. From the stress – strain curves we can infer that the results for user – defined elements are in agreement with both the Abaqus Element as well as the analytical solution

Having verified that the implementation of the 9-node UEL with the traditional formulation as well as with the mixed formulation is correct, we have completed the first batch of verification simulations. Abaqus [2] advises users to verify the implementation using one finite

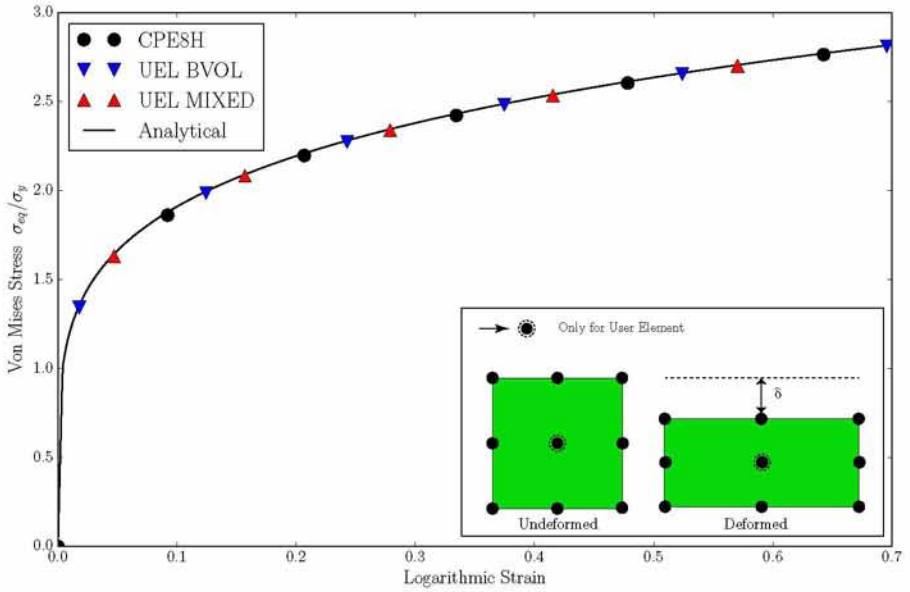


Fig. 5.8: The stress (Von Mises) - strain (equivalent Logarithmic strain) curves during uniaxial tension with displacement control, using user–defined elements and Abaqus elements along with the curve for the analytical solution in this case

element with prescribed traction or boundary condition and verify the results with Abaqus elements. Having done so, the next step would be to gradually increse the complexity of the problem in order to determine error in our implementation (if any).

Before we proceed to the next batch of simulations conducted in the context of this thesis we have to address a subtle issue regarding force control simulations vs displacement control simulations. In order to take into account force control simulations, where forces (concentrated or distributed) are applied to one or more surfaces of the finite element, we need to change the signs of the residual vector (RHS) and of the stiffness matrix (AAMATRX). For more information about this change in signs you may refer directly to the user subroutine in Appendix C of this thesis.

5.2.iii Contact analysis

In this section, we are interested in examining the case of contact between one element and a rigid body. Adding one more degree of complexity to the problem, we introduce the problem of deforming a finite element by indirectly applying a force field through contact. For this purpose, we used the 9 – node (2D), 2^{nd} order Lagrangian finite elements with/without linear pressure and Abaqus elements [4]. Contact problems solved by Abaqus Implicit [2] are in general very difficult and in many cases impossible to solve. Since however the user element was developed to accurately calculate stresses and displacements in the case of finite deformation metal plasticity (such as deformation processes) it is important to validate that

[4] In particular, we used the CPE8H finite elements

the UEL subroutine can also handle problems that involve hard contact.

The undeformed and deformed configuration of the one element used in the contact analysis is shown in Figure 5.9 along with the rigid die.

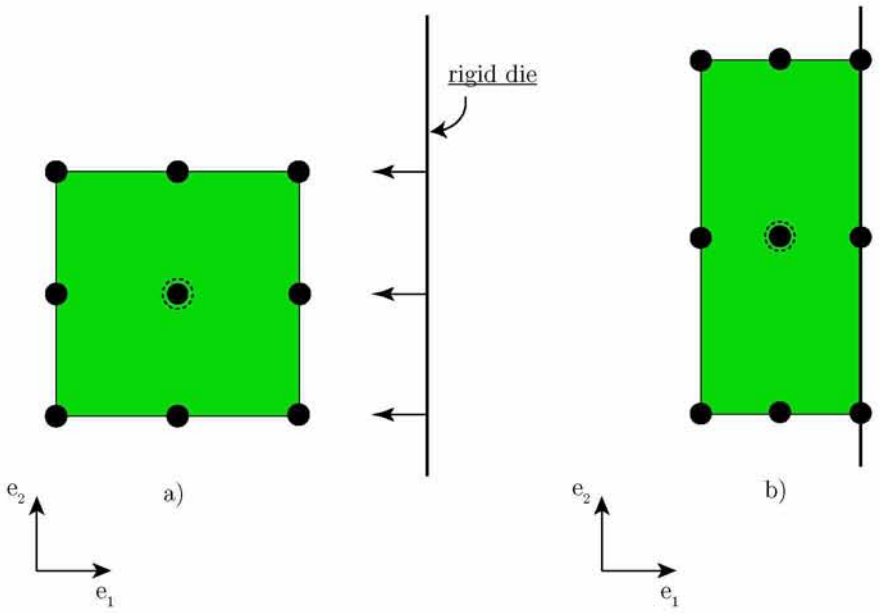


Fig. 5.9: The undeformed (a) and deformed (b) meshes in the case of a single finite element under uniaxial compression through hard contact with a rigid die. The black points indicate the location of each node in the finite element. The central node only exists in the case of User ELements. The vertical solid black line represents the rigid die that moves in the $\mathbf{e}_1$ *direction and deforms the finite element. The boundary conditions are such to eliminate rigid body translations and rotations for the finite element*

During contact analysis, the rigid die was displaced in the $\mathbf{e}_1$ directon. Once the contact between the die and the element was initiated, the contact forces were able to deform the element as the die was pushing against it. We report the results of this simulation in the sense of a stress (Von Mises) – strain (equivalent Logarithmic Strain) curve which is shown in figure 5.10.

From the stress – strain curve shown in figure 5.10 we can see that the results from contact analysis with user – defined elements and Abaqus elements are coincide with analytical solution of the problem, which means that UEL subroutine works properly and in this case.

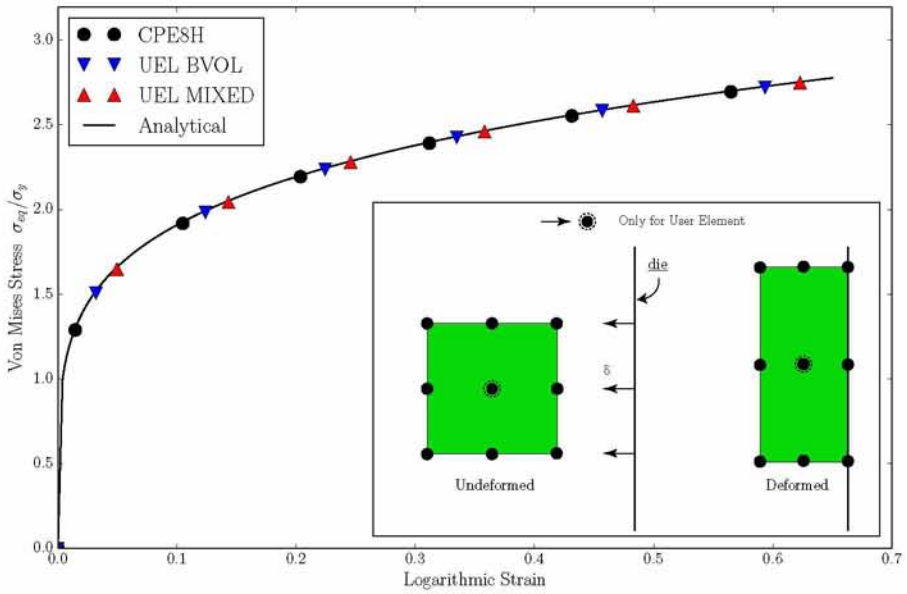


Fig. 5.10: The stress (Von Mises) - strain (equivalent Logarithmic strain) curves during uniaxial compression through hard contact with a rigid die, using user–defined elements and Abaqus elements along with the curve for the analytical solution in this case

5.3 Wire drawing

5.3.i General Overview and Problem Description

Wire drawing is one of the most used metal forming processes that involves plastic deformation. During this metal working[5] process a billet is pulled through a die, or series of dies, so that to reduce its cross–sectional area. In the case of cylindrical wires, the reduction of the cross sectional area is equivalent to the reduction of the cross section's radius. As the wire is pulled through the die and deforms plastically, its volume remains constant and as a result the diameter decreases while the length increases. There are many applications for wire drawing products including but not limited to electrical wiring, cables, tension–loaded structural components etc. Drawing is usually performed at room temperature, and therefore is classified as a cold working process. The main variables involved in this process comprise of the die semiangle, α, the cross-sectional area A, the friction coefficient μ, the reduction area r and the yield tension σ_y.

In order to solve this problem numerically, we made several assumptions regarding material properties and process conditions. The problem is treated from a continuum mechanics point of view which means that metallurgical variables such as grain size, internal defects like dislocations and structural inhomogeneities are neglected in this problem. In regards to the friction coefficient, we assumed frictionless constant between the billet and the rigid die since the process usually takes place in a well-lubricated environment. While accounting for

[5]Is a process of working with metals to create individual parts, assemblies or large–scale structures

friction would greatly complicate the problem, it would not contribute significantly to its solution. Friction gives rise to a temperature increase during the process, but in reality the assembly is constantly cooled by a lubricant and specific cooling fluids that also keep the friction coefficient relatively low.

5.3.ii Finite Element formulation of wire drawing using Abaqus elements

In this section we discuss the finite element implementation of the wire drawing process outlining the model's details and any assumptions made in modelling process. The billet [6] was modelled as an axisymmetric body/wire with an initial radius equal to $R_0 = 1.26mm$ [7]. The die was modelled as a rigid body with a semi–angle equal to 4^o degrees. The reduction region, r, of the die was determined by equation $r = \frac{R_0 - R}{R_0}$ [8]. The wire was taken to be made of steel with thin zinc coating[9]. In this particular problem, the steel extended from $0 - 0.972R_0$ whereas the zinc coating was taken to be the remaining $0.028R_0$ of the total radius R_0. A schematic representation of geometry is shown in figure 5.11.

Figure 5.12(a), shows a snapshot of the finite element mesh used for this problem along with the rigid die that causes the wire to deform when the drawing process is active. The figure also depicts a detail of the finite element mesh indicating the finite elements comprising the zinc coating. The mesh consists of 2 dimensional, 2nd order axisymmetric finite elements (CAX8H) from the Abaqus library [2]. The details of the finite element mesh are outlined in table 5.1 that follows.

Table 5.1: The finite element mesh used to discretize the problem of wire drawing of a steel wire with a thin zinc coating

Elements	Nodes	d.o.f.
1350	5555	12666

In particular, 27 elements were used in r – direction, 2 of them being elements used to model the zinc coating, while on z – direction we used 50 elements. These values were decided after a corresponding mesh refinement study. Figure 5.12(b), shows the 3-dimensional equivalent problem that we are actually modelling. Taking advantage of the cylindrical symmetry, we only have to model the axisymmetric part of this problem which not only involves considerably fewer degrees of freedom but is also much easier to solve in terms of modelling

[6] The deformable body

[7] The data for this problem are actual data used in the wire drawing process by SYRMA S.A., Volos, Greece

[8] In our problem the reduction region was set equal to $0.06mm$ as the initial radius was determined equal to 1.26mm while the final radius was determined equal to 1.18mm

[9] In most cases of metal working processes, billets are used to have coatings in order to have higher resistance on the forces that the dies apply

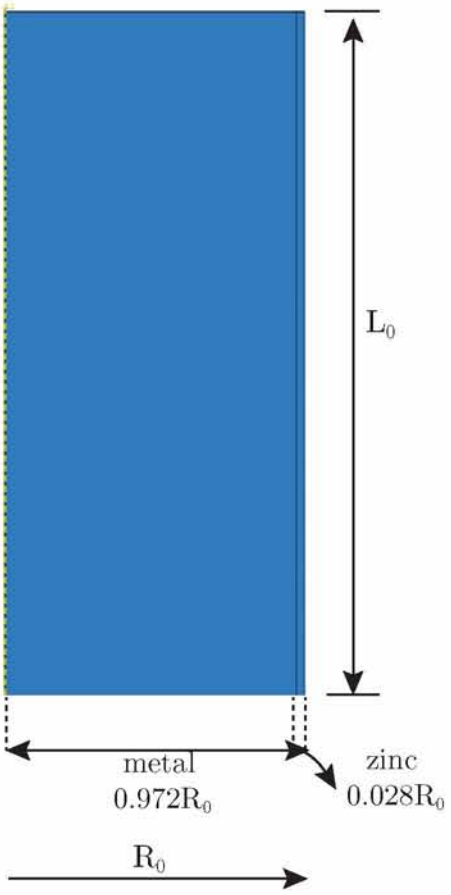


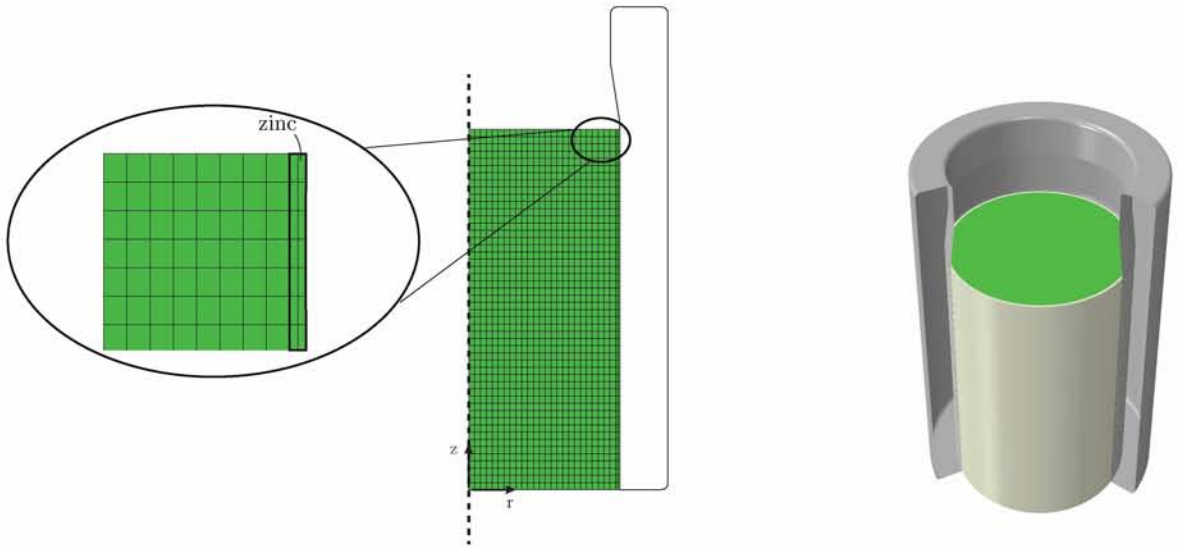
Fig. 5.11: A schematic representation of the axisymmetric wire made of steel with a thin zinc coating. The wire is subjected to drawing in order to reduce its cross-sectional area

the contact interactions [10]. However, Abaqus [1] has the capability of projecting the axisymmetric results in the full 3D geometry which makes the results more visually appealing but also more comprehensible.

The billet is subjected to a uniform displacement field applied on the nodes of the upper surface forcing the wire to pass through the die region. From our simulations, we found that a displacement of $4L/3$ was capable of leading the billet out of the rigid die with the desired reduction in radius. This, displacement field imposed in two steps. On the first step, the billet moved only 0.05mm to initiate contact. On the second step, we impose the remaining displacement and plastic deformation takes place.

The material properties used for the metal and the zinc coating on billet are summarized below.

[10] Modelling the 3-dimensional contact interaction in the case of the full 3-d problem would be almost impossible. Such a problem would require a very fine mesh which would make the analysis very challenging even for modern computers



(a) (b)

Fig. 5.12: The finite element mesh used to discretize the model's geometry (a) along with a detail illustrating the region consisting of zinc elements. The mesh consists of 2nd order hybrid axisymmetric elements from the Abaqus library [2]. The 3 dimensional equivalent problem is shown in (b) where part of the rigid die is removed for visualization purposes. Steel is coloured green whereas grey is used for the zinc coating. The die is coloured dark grey and is assumed to be rigid

Properties used for steel:

- Young's modulus,E, 216 GPa
- Poisson's ratio, ν, 0.295
- Yield Stress, σ_0, 850 MPa
- Work Hardening law that follows Hollomon's curve $\sigma = \sigma_0 + k \cdot \varepsilon^n$, where n=0.16 and k=978.7808 MPa

Properties used for the zinc coating:

- Young's modulus,E, 107 GPa
- Poisson's ratio, ν, 0.33
- Yield Stress, σ_0, 166 MPa
- Work Hardening law that follows Hollomon's curve $\sigma = \sigma_0 + k \cdot \varepsilon^n$, where n=0.225 and k=372.317 MPa

Case 1: The wire is pulled through one die

We will now present the finite element results in the case where only one die used in the simulation. Figure 5.13 and figure 5.14 show the contour plots of the distribution of the Von Mises equivalent stress and pressure in the billet, respectively. We will limit our discussion to the contours of the 2D configuration as they are equivalent with the 3D, due to the symmetry of the problem.

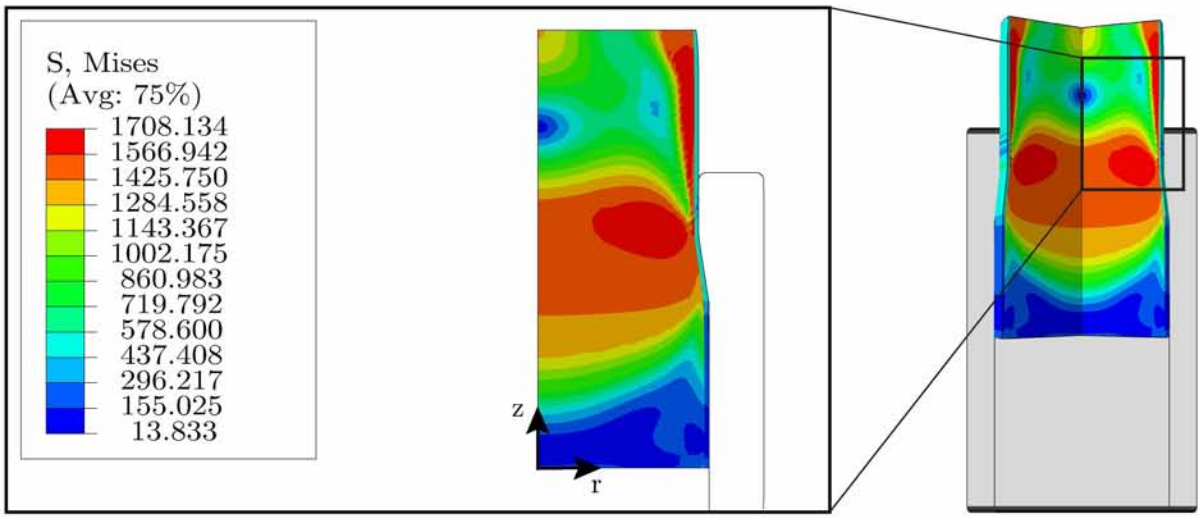


Fig. 5.13: A contour plot of the Von Misses equivalent stress while the wire is passing through the rigid die

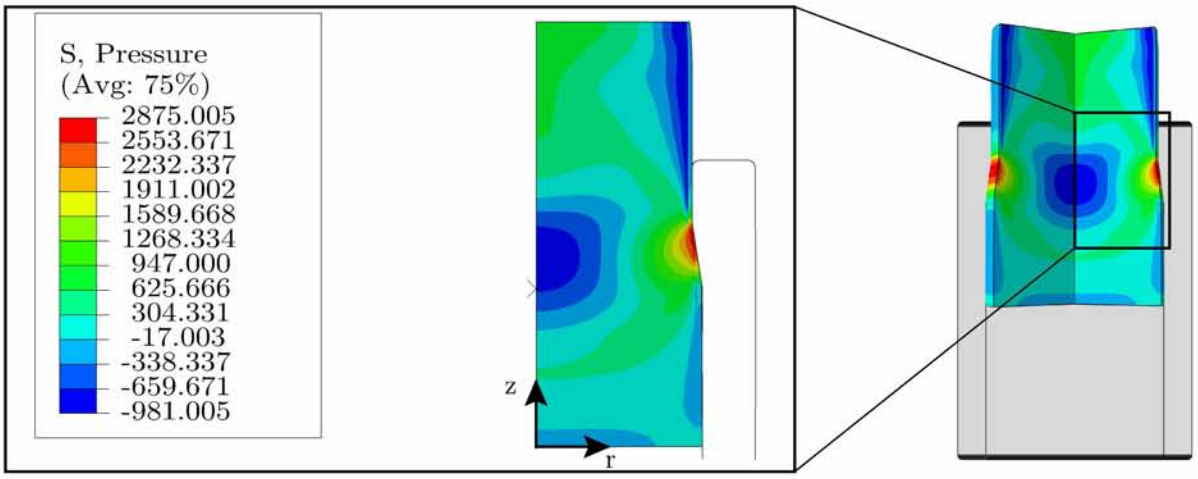


Fig. 5.14: A contour plot of the pressure in the wire while the wire is passing through the rigid die

As expected, the maximum value of the Von Mises equivalent stress occurs inside the die. This is reasonable if we take into account that inside the die the billet is subjected to very high forces from the die in order to change its length and radius. Moreover, regarding the contours of the Von Mises stress we can conclude that the distribution inside the die is uniform which is an indication that entrance effects are no longer present. In reality the

wire should be modelled as an infinitely long deformable axisymmetric body, but this would not be realizable from a finite element point of view. Therefore, we have to limit the length of the wire in order to reduce the computational cost. However, upon entering and exiting the die, the wire undergoes a state of deformation which is not representative of the actual problem. The real wire drawing process takes place in steady state conditions and therefore the wire in our simulations has to be long enough to capture a region of uniform stress. This seems to be the case in our problem as clearly there is a region where the stresses are uniformly distributed with respect to the z direction. as the process is in a steady – state in this region. If we had the opportunity to simulate a wire with infinite length, then we could see that the distribution of the Von Mises equivalent stress would be uniform along the z – direction. The aforementioned statement also explains why the distribution of the Von Mises stress is nonuniform in the region where one part of the billet has not entered the die yet and in the region where a part of the billet has pulled out of the die.

Now, concerning the pressure contours shown in figure 5.14, we can report that inside the die we have two different regions regarding the value and the sign of the pressure. The region next to the die has the maximum value of pressure and a positive sign, while the region next to the axis of symmetry (on the left) has the lowest value of pressure and a negative sign. However, we have to note that on Abaqus the value of the pressure is given through equation $p = -\sigma_{kk}/3$. Thus, the values next to the die have actually negative sign (compressive) while the values next to the axis of symmetry have a positive sign.

Figure 5.15, shows a plot of the reaction force in z – direction versus the displacement field applied in z – direction. We can see clearly from the same figure that the first time the billet enters the die the reaction force obtains the maximum value. After a period of time the reaction force reaches a steady – state and is eventually equal to zero when the billet has pulled completely out of the die.

Figure 5.16, shows the radial distribution of the axial stress, σ_{zz}. From the plot we can infer that σ_{zz} remains compressive along most of the radius of the billet and becomes tensile only very close to the die (outer surface). The aforementioned statement can be explained by taking into account that each cross – section in the steady – state residual stress region should transmit zero total axial force.

Figure 5.17, shows the distribution of hydrostatic stress component, $\sigma_{kk}/3$ normalized by the tensile yields stress σ_0 at the central surface, when the billet is pulled and forced to enter the die. We can infer that σ_{kk} is tensile near the entrance of the reduction region, along the reduction region and in a small region after the exit of the die, whereas it is compressive in all other regions.

Case 2: The wire is pulled through two dies

In this case, the wire was pulled through two consecutive dies. The geometry and the finite element mesh of the billet along with the material properties are the same as the ones

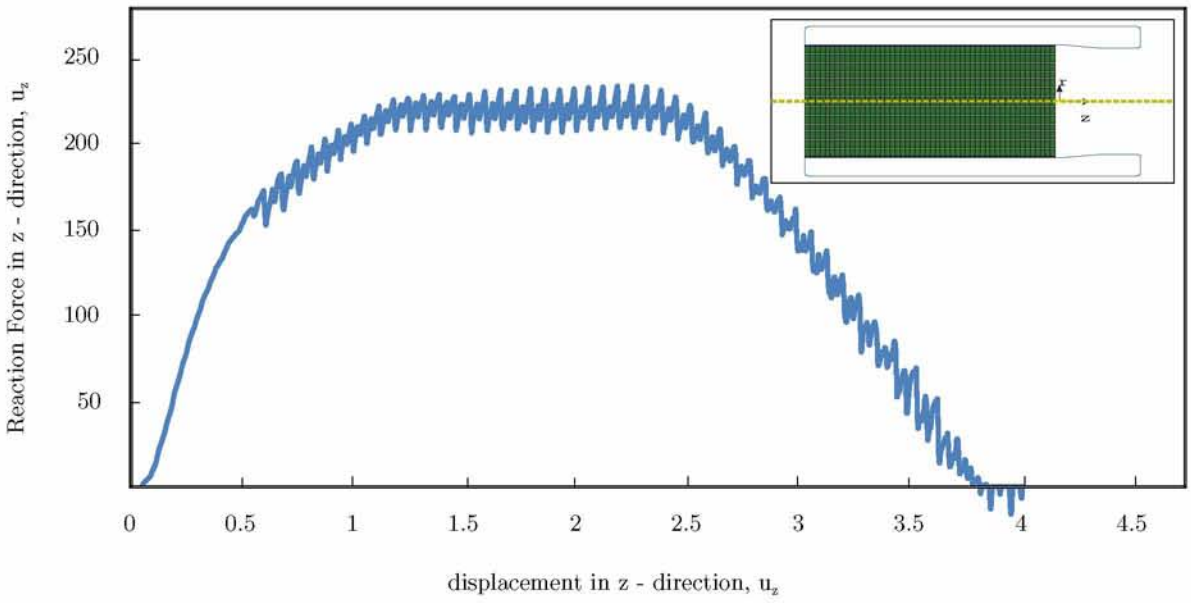


Fig. 5.15: Plot of the reaction force in z – direction versus the applied displacement field in z – direction

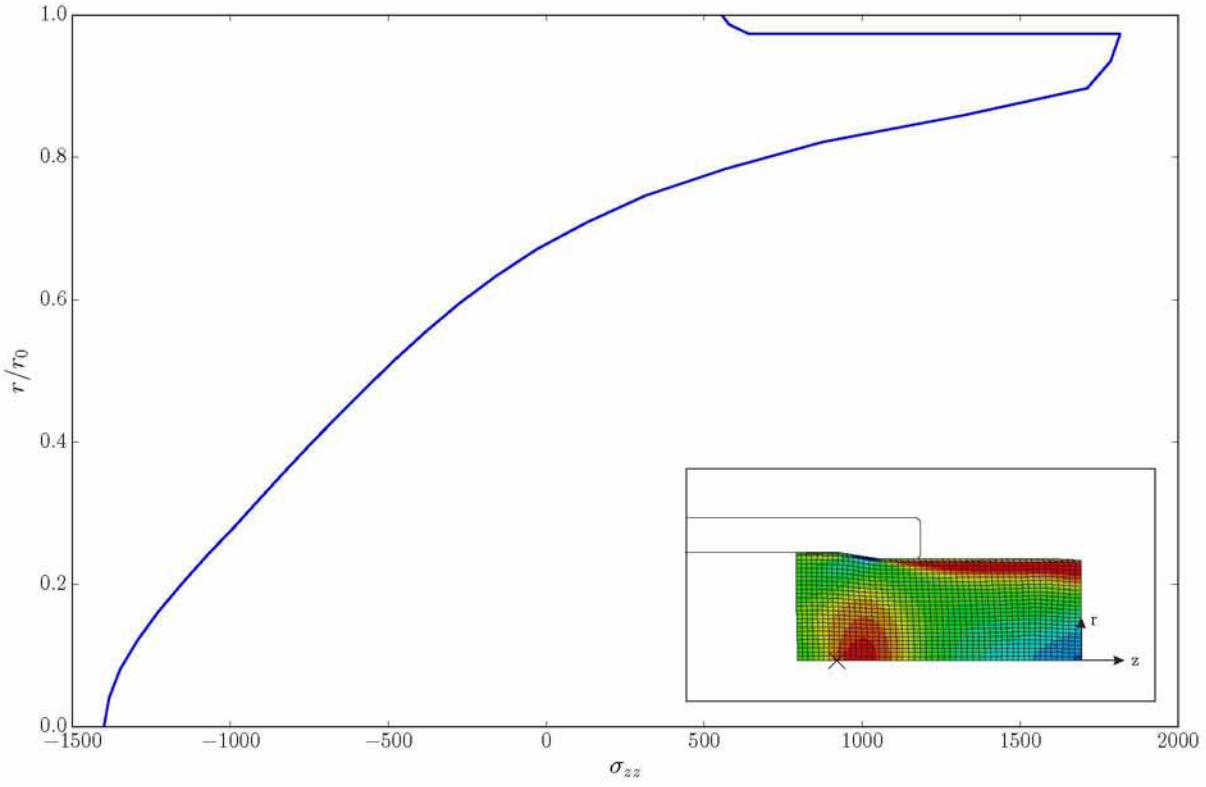


Fig. 5.16: Plot of the radial distribution of the axial residual stress σ_{zz}

outlined in the previous section. The reduction region of the 1^{st} die was set equal to r_1 and the semi – angle α_1, while on the 2^{nd} die the reduction region was set equal to r_2 and the

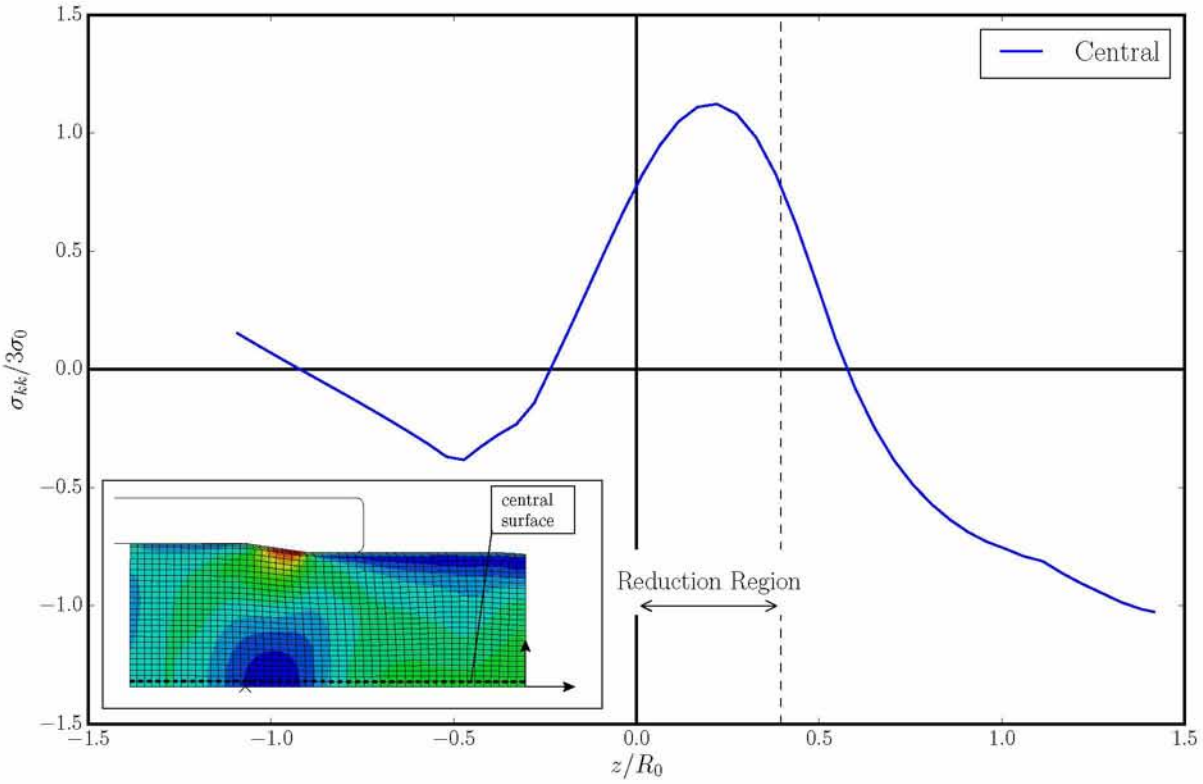


Fig. 5.17: The distribution of $\sigma_k k$ stress along the central surface of the wire

semi – angle α_2 [11].

In figure 5.18, we present the finite element mesh along with the two dies that were used in the Abaqus simulations. A 2D and a 3D configuration are shown on each figure. The 2D configuration is equivalent with the 3D one, due to the symmetry of the axisymetric wire. Thus, we discuss the results based on the 2D configuration. In all contour plots we focus on an area where one part of the billet is still inside the die.

Figures 5.19 and 5.20, show the contour plots of the Von Mises equivalent stress along the billet in the 1^{st} and the 2^{nd} die respectively. The maximum values of the Von Mises equivalent stress occur when the wire is still inside the two dies in this case as well. This is a reasonable result if we take into account that the die applies extremely high forces on the billet in order to deform the wire to its final desired radius. Moreover, regarding the contours showing the Von Mises stress we can conclude that the distribution inside the die is uniform as the process is in a steady – state in this region as mentioned previously in the case of wire drawing with one die.

Now, we proceed with the contours showing the distribution of the pressure along the billet. From figures 5.21 and 5.22 we can conclude that the higher values of pressure occur inside

[11] We used on the 1^{st} die an initial radius R_0^1=1.26mm, a final radius R_1^1=1.181985mm and a semi – angle $\alpha_1 = 4^o$, while on the 2^{nd} die we used as initial radius the $R_1^1 = R_0^2$=1.181985mm, a final radius R_1^2=1.1088mm an a semi – angle $\alpha_2 = 4.5^o$.

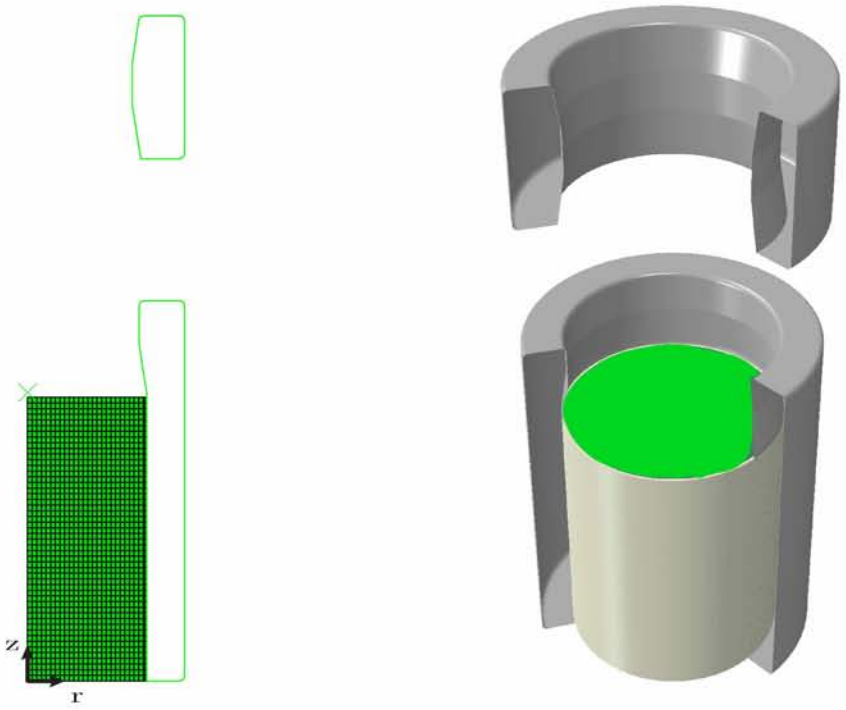


Fig. 5.18: The finite element mesh used to discretize the model's geometry (a) along with a detail illustrating the region consisting of zinc elements. The mesh consists of 2nd order hybrid axisymmetric elements from the Abaqus library [2]. The 3 dimensional equivalent problem is shown in (b) where part of the rigid die is removed for visualization purposes. Steel is coloured green whereas grey is used for the zinc coating. The died are coloured dark grey and is assumed to be rigid

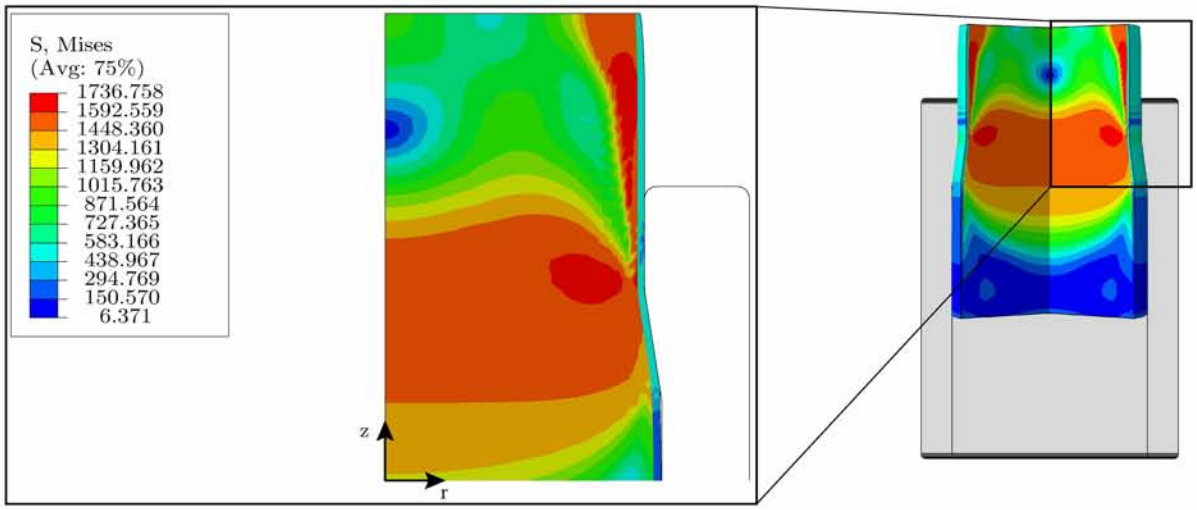


Fig. 5.19: The distribution of the Von Mises equivalent stress on the billet when the latter is still inside the first die

the die and more specifically next to the die (on the right region). The high compressive forces that the dies apply through the contact with the billet can explain the high values of pressure occuring inside the die.

Figure 5.23 shows the reaction force in z – direction versus the displacement field applied on

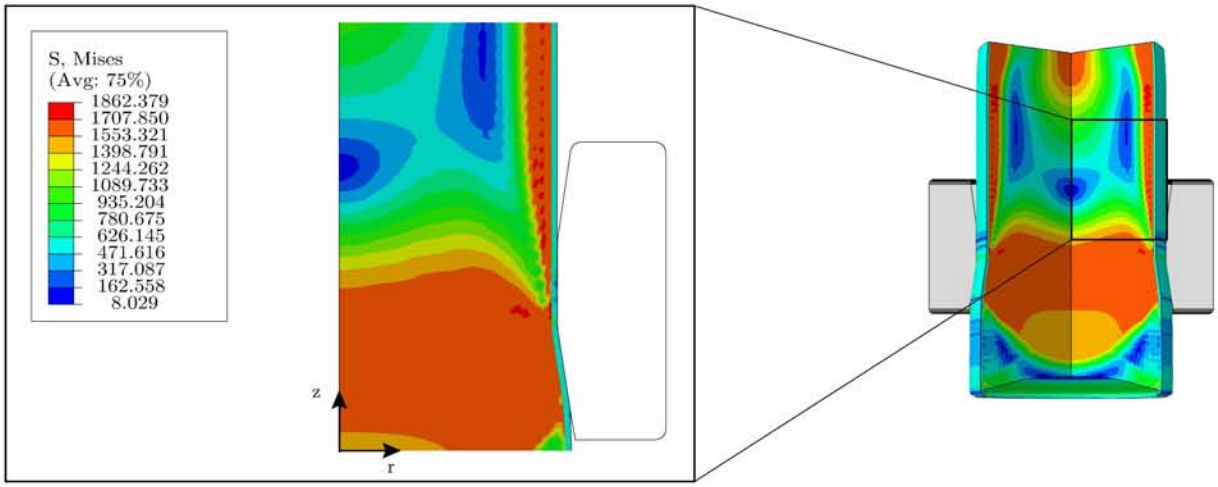


Fig. 5.20: The distribution of the Von Mises equivalent stress on the billet when the latter is still inside the first die

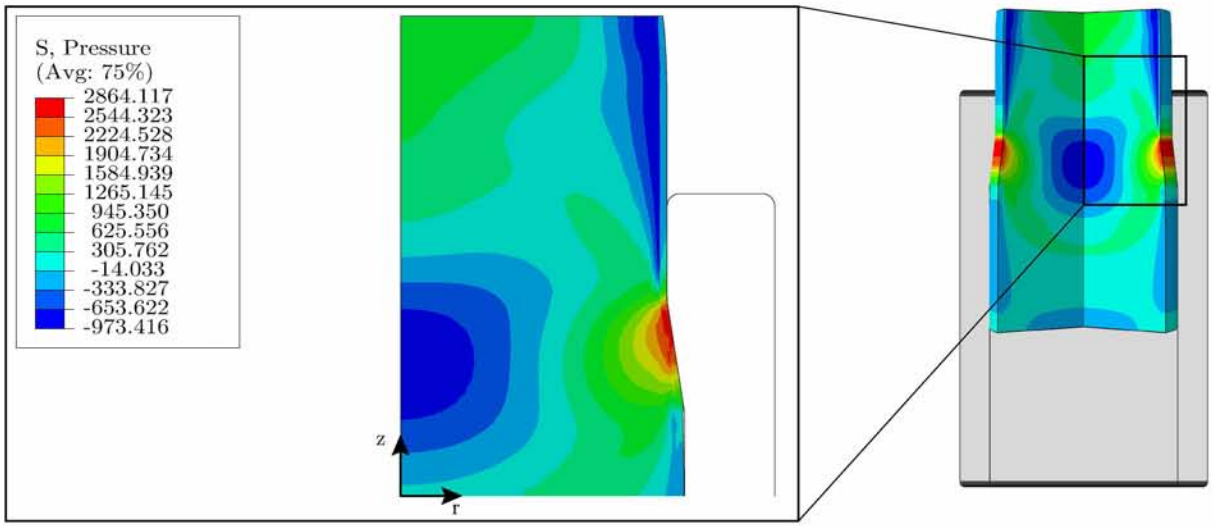


Fig. 5.21: The pressure distribution on the billet when the latter is still inside the first die

the same direction. It is obvious that, in both two dies, the reaction force reaches a steady – state value. In the same figure, 5.23 we can see a peak in the reaction force. This peak in the reaction force occurs when the billet is entering the 2^{nd} die where it encounters very high forces from both dies. Finally, when the wire is pulled completely out of the die, then the reaction force drops again to zero.

Figures 5.24 and 5.25 show the radial distribution of the axial stress in the case where the billet has entered the 1^{st} and the 2^{nd} die respectively. We can see that σ_{zz} remains compressive along most of the radius of the billet and becomes tensile only near the region where is the die (outer surface). The aforementioned statement can be explained by taking into account that each cross – section in the steady – state residual stress region should transmit zero total axial force.

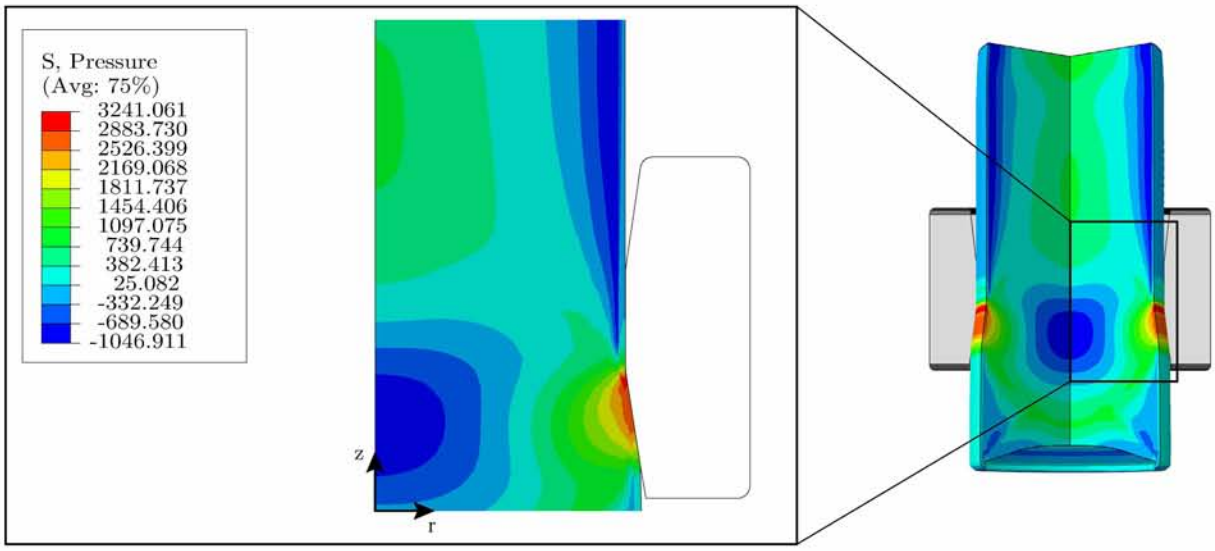


Fig. 5.22: The pressure distribution on the billet when the latter is inside the second die

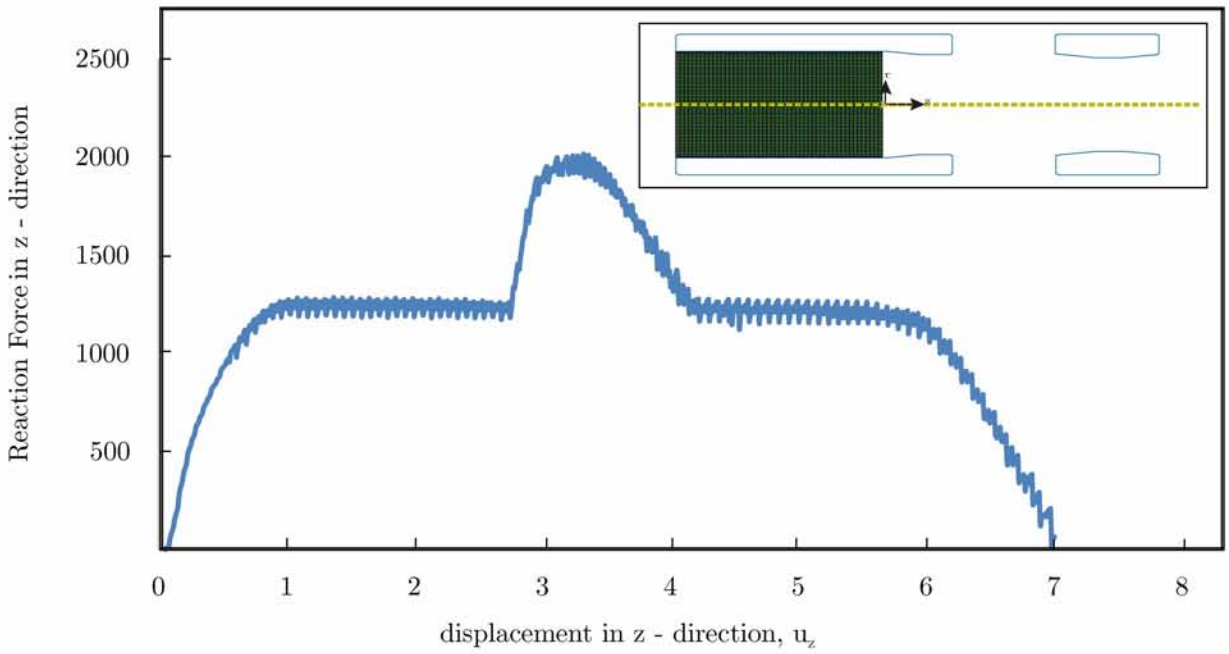


Fig. 5.23: Plot of reaction force in z – direction versus the applied displacement field in z – direction

Figures 5.26 and 5.27 show the distribution of hydrostatic stress, $\sigma_{kk}/3$ normalized by the tensile yields stress σ_0 at the central surface, when the billet is pulled from the 1^{st} and the 2^{nd} die respectively. From figure 5.26 we can see that σ_{kk} is tensile near the entrance to the reduction region and along the reduction region while the σ_{kk} component is everywhere else compressive. Now, during the pulling from the 2^{nd} die the σ_{kk} component is tensile everywhere except from the area near the reduction region (or the entrance of the die) where σ_{kk} it is compressive.

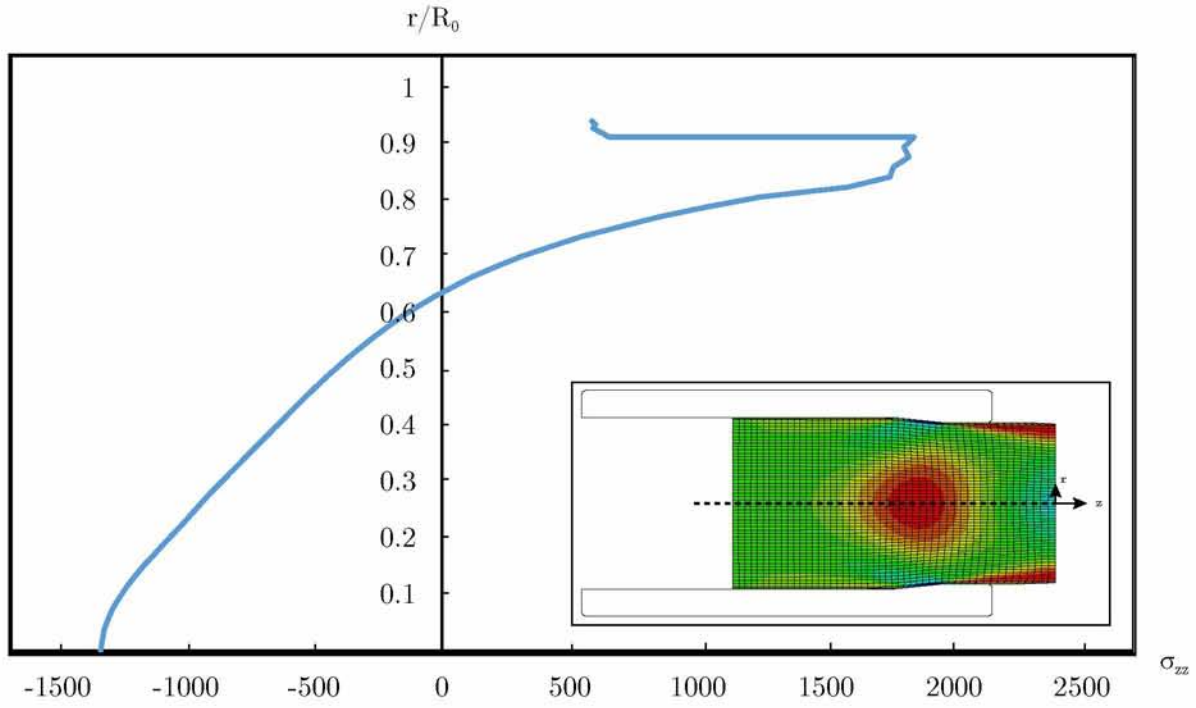


Fig. 5.24: Plot of the radial distribution of the axial residual stress σ_{zz} during the pass from the 1^{st} die

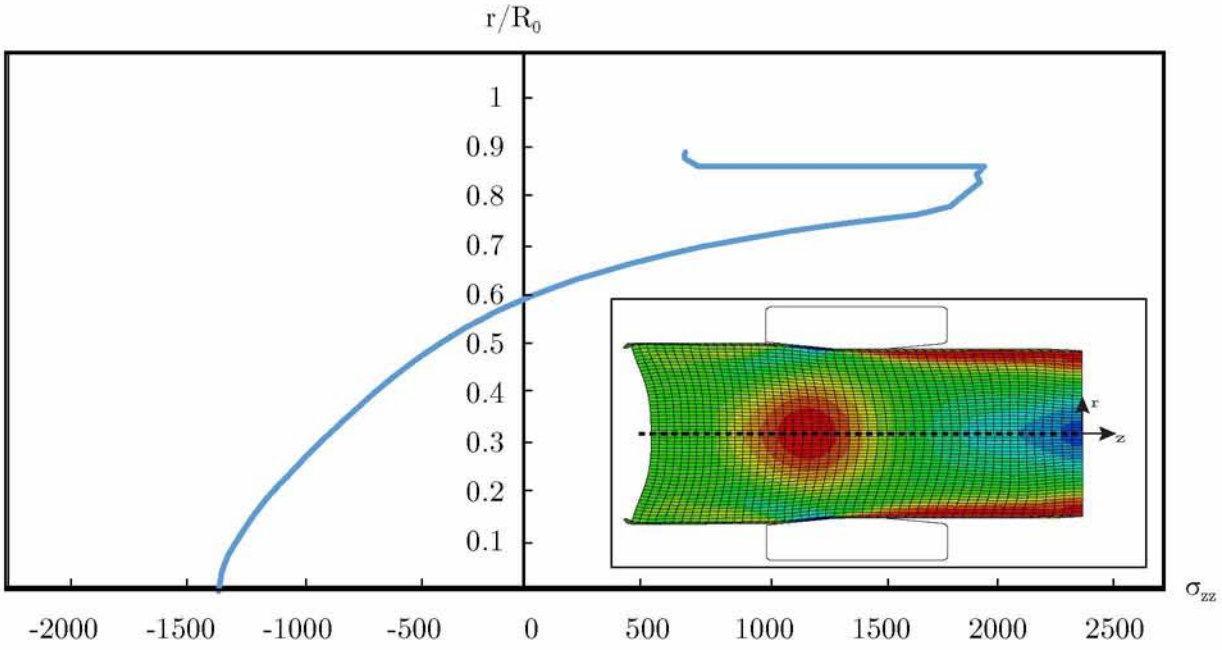


Fig. 5.25: Plot of the radial distribution of the axial residual stress σ_{zz} during the pass from the 2^{nd} die

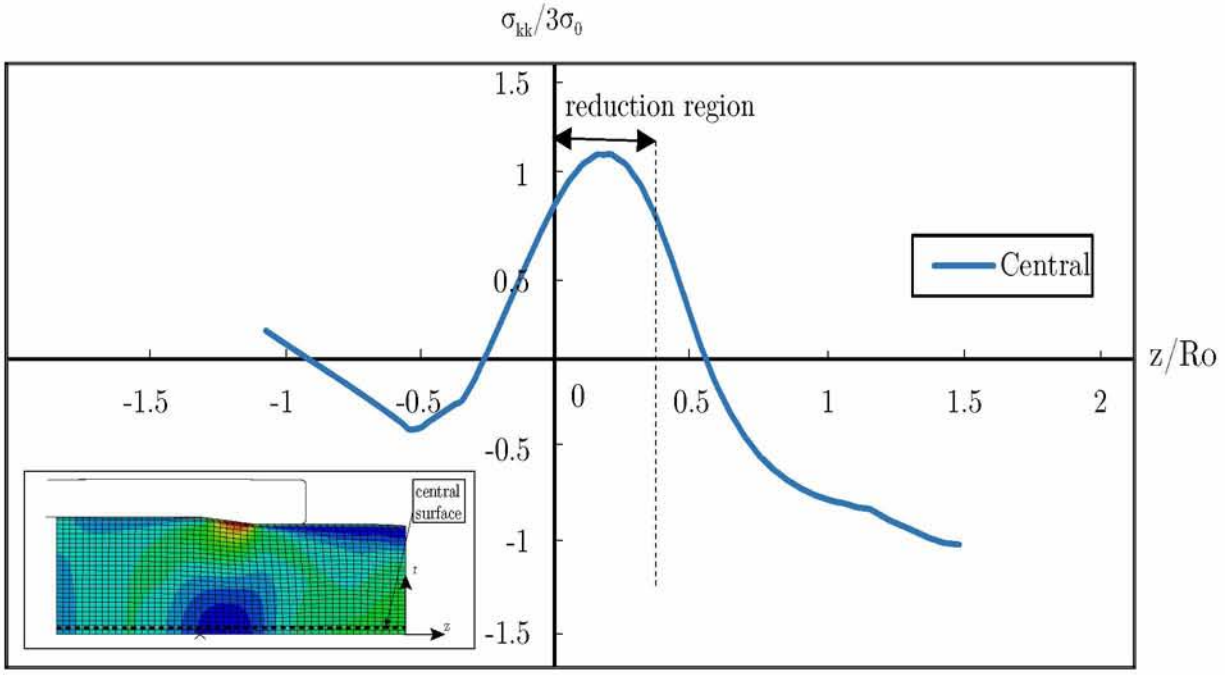


Fig. 5.26: The distribution of $\sigma_k k$ stress at steady – state while the wire is inside the 1^{st} die

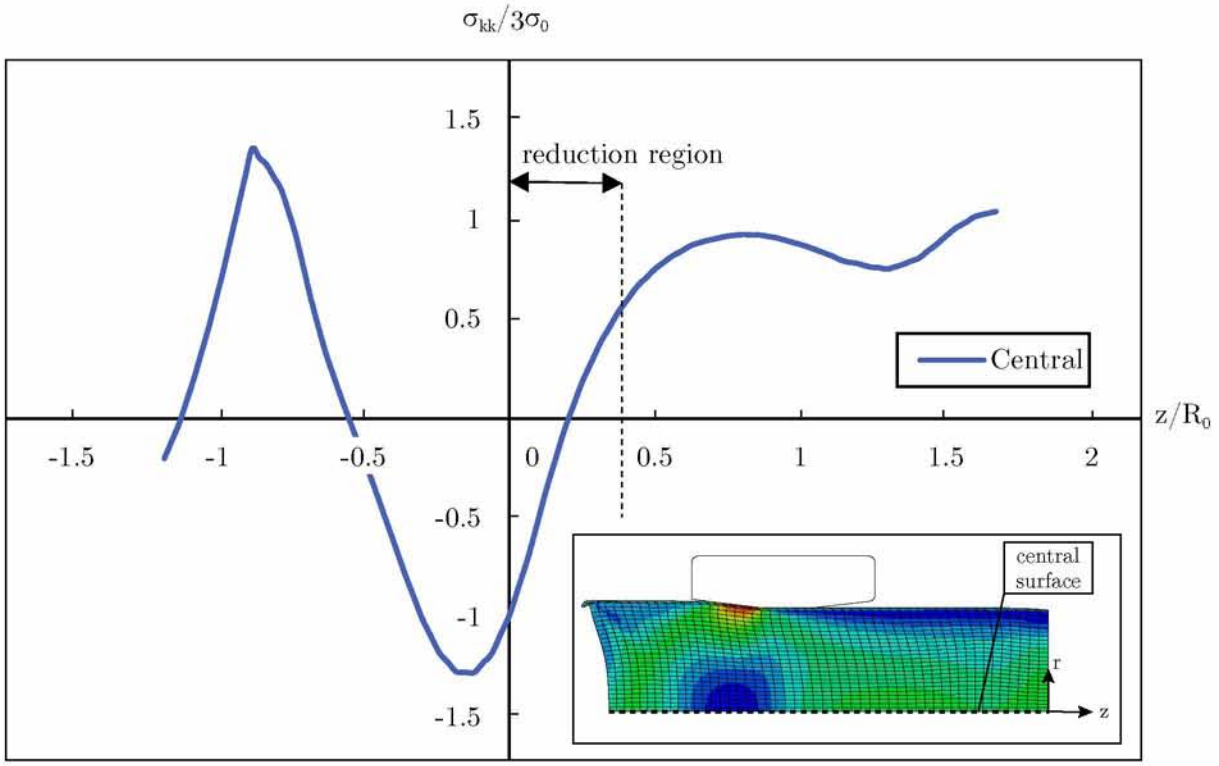


Fig. 5.27: The distribution of $\sigma_k k$ stress at steady – state while the wire is inside the 2^{nd} die

Bibliography

[1] ABAQUS/Standard, Version 6.12 © Dassault Systèmes', (2012)

[2] ABAQUS, 'Abaqus Theory Manual, Version 6.12 © Dassault Systèmes', (2012)

[3] Aravas N., 'On the numerical integration of a class off pressure – dependent plasticity models', *Int. J. Numer. Methods Eng.*, (1987): 1395 – 1416

[4] Aravas N., 'Finite Elastoplastic Transformations of Trasversely Isotropic Metals', *Int. J. Solids Struct.*, **29**, (1992): 2137–2157

[5] Aravas N. (2005), 'Cartesian Tensors',University of Thessaly Publications

[6] Aravas N. (2014), 'Mechanics of Materials Volume I: *An Introduction to the Mechanics of Deformable bodies and Linear Elasticity*' Tziolas Publications

[7] Aravas N. and Aifantis E.C., 'On the Geometry of Slip and Spin in Finite Plastic Deformation', *Int. J. Plast.*, **7**, (1991): 141–160

[8] Asaro R.J. and Lubarda V.A. (2006), 'Mechanics of Solids and Materials',Cambridge University Press

[9] Belytschko Ted, Wing Kam Liu, Brian Moran, Khalil I. Elkhodary (2014), 'Nonlinear Finite Element for Continua and Structures', 2^{nd} Edition WILEY

[10] Bigoni D. (2012), 'Nonlinear Solid Mechanics', Cambridge University Press

[11] Bonet J. and Wood R.D. (2008), 'Nonlinear Continuum Mechanics for Finite Element Analysis',Cambridge University Press

[12] Fionn Dunne, Nick Petrinic (2005), 'Introduction to Computational Plasticity', OXFORD UNIVERSITY PRESS

[13] Gurtin M.E.,Fried E. and Annand L., (2009) 'The Mechanics and Thermodynamics of Continua',Cambridge University Press

[14] Hill R. (1950), 'The mathematical theory of plasticity', Oxford University Press

[15] Hjelmstad K.D. (2005), 'Fundamentals of Structural Mechanics',2nd edition Springer, Academic Press

[16] Johnson W. and Mamalis A. G. (1977), 'Engineering Plasticity: Theory of Metal Forming Processes', Vol.2

[17] Koiter W.T., 'Stress-strain relations, uniqueness and variational theorems for elastic-plastic materials with a singular yield surface', *Q. Appl. Math.*, **11**, (1953): 350–354

[18] Lubliner J. (2006), 'Plasticity Theory', University of California at Berkeley

[19] Lawrence E. Malvern 1969, 'INTRODUCTION TO THE MECHANICS OF A CONTINUUM MEDIUM',Michigan State University

[20] Lee E.H., 'Elastic–Plastic Deformation at Finite Strains', *J. Appl. Mech.*,**36(1)**, (1969): 1–6

[21] Lubarda V.A., 'Constitutive theories based on the multiplicative decomposition of deformation gradient: Thermoelasticity, elastoplasticity, and biomechanics', *Appl. Mech. Rev.*,**57(2)**,(2004):95–108

[22] Lubliner Jacob (2006), 'PLASTICITY THEORY', University of California at Berkeley

[23] Malvern L.E. (1969), 'Introduction to the Mechanics of a Continuous Medium', Prentice Hall

[24] Mandel J., 'Une gènèralisation de la theoriè de la plasticitè de W.T.Koiter', *Int. J. Solids Struct.*, **1**, (1965) : 273–295.

[25] Needleman.A., 'On the finite element formulations for large elastic-plastic deformations', *Comput. and Struct.*,**20**, (1985): 247–257

[26] Papatriantafyllou I.C., 'Trip Steels: Constitutive Modeling and Computational Issues', *University of Thessaly, School of Engineering, Department of Mechanical Engineering*, Ph.D. Thesis, (2005)

[27] Taylor G.I., 'Plastic Strain in Metals', *J. Inst. Metals*,**62**, (1938): 307–324

[28] Tomlenov A.D. (1951), 'Theory of Plastic Deformation in Metals', Mashgiz, Moscow

[29] Thompsen E.G., Yang C.T. and Kobayashi S. (1956), 'Mechanics of Plastic Deformation in Metal Processing', MacMillan, N.Y.

[30] O.C. Zienkiewicz, R.L. Taylor (2000), The Finite Element Method, 5^{th} edition, Volume II: Solid Mechanics OXFORD AUCKLAND BOSTON JOHANNESBURG MELBOURNE NEW DELHI

Appendices

Appendix A

Input File for the simulation of wire drawing with 2 dies

```
*HEADING
 DRAWING
*RESTART,WRITE,FREQ=100
*INCLUDE,INPUT=GEOM.inp
*MATERIAL,NAME=METAL
*ELASTIC
216000.,0.295
***
***     Input files for metal plasticty using Hollomon's curve
***
*INCLUDE,INPUT=metal_plasticity.inp
**
*SOLID SECTION,ELSET=ALLE,MATERIAL=METAL
*MATERIAL,NAME=ZINC
*ELASTIC
107000.,0.33
***
***     Input files for zinc coating plasticity using Hollomon's curve
***
*INCLUDE,INPUT=zinc_plasticity.inp
**
*SOLID SECTION,ELSET=OVERLAP,MATERIAL=ZINC
**
```

```
***
***     Constructing the 2 rigid dies
***
*SURFACE,TYPE=SEGMENTS,NAME=RIGID1,FILLET RADIUS=0.055
START ,1.26 ,-3.0500000
LINE ,1.26 ,0.0000000
LINE ,1.181985 ,0.4987090
LINE ,1.181985 ,0.9715030
LINE ,1.66 ,0.9715030
LINE ,1.66 ,-3.050000
LINE ,1.26 ,-3.050000
*RIGID BODY,ANALYTICAL SURFACE=RIGID1,REF NODE=100000
*ELSET,ELSET=CONTACT,GENERATE
27,1350,27
*SURFACE,NAME=OUT1,TYPE=ELEMENT
CONTACT
*CONTACT PAIR,INTERACTION=ONE
OUT1,RIGID1
*SURFACE INTERACTION,NAME=ONE
***
***
***
*SURFACE,TYPE=SEGMENTS,NAME=RIGID2,FILLET RADIUS=0.055
START ,1.2  ,2.4715030
LINE ,1.181985 ,2.5715030
LINE ,1.1088  ,3.0393330
LINE ,1.1088  ,3.4828530
LINE ,1.1819850,  3.9828530
LINE    ,   1.66     ,  3.9828530
LINE    ,   1.66     ,  2.4715030
LINE    ,   1.2      ,  2.4715030
*RIGID BODY,ANALYTICAL SURFACE=RIGID2,REF NODE=100001
*SURFACE,NAME=OUT2,TYPE=ELEMENT
CONTACT
*CONTACT PAIR,INTERACTION=TWO
OUT2,RIGID2
*SURFACE INTERACTION,NAME=TWO
**
**FRICTION,TAUMAX=0.577
**0.1
**
**
```

```
*BOUNDARY
LEFT,1,1
100000,1,6
100001,1,6
**
*EQUATION
2
TOP,2,1.,200000,2,-1.
**
***
***     Steps during simulation
***
*STEP,INC=50,NLGEOM
*STATIC
0.01,0.05,,0.01
*CONTROLS,PARAMETERS=FIELD,FIELD=DISPLACEMENT
5.D-3
*BOUNDARY
PULL,2,2,0.05
*END STEP
***
***
***
*STEP,INC=10000,NLGEOM
*STATIC
0.005,3.95,,0.005
*CONTROLS,PARAMETERS=FIELD,FIELD=DISPLACEMENT
5.D-3
*BOUNDARY
PULL,2,2,4.
**
*EL PRINT,ELSET=CONTACT,POSITION=CENTROIDAL,FREQUENCY=200
PE
**
**
*OUTPUT,FIELD,FREQUENCY=1,VARIABLE=PRESELECT
*ELEMENT OUTPUT
S,PE
*NODE OUTPUT
U
*OUTPUT,HISTORY,FREQUENCY=1
*NODE OUTPUT,NSET=PULL
```

```
RF,U2
*NODE FILE,FREQUENCY=1,NSET=PULL
RF
*END STEP
***
***
***
*STEP,INC=10000,NLGEOM
*STATIC
0.005,3.,,0.0005
*CONTROLS,PARAMETERS=FIELD,FIELD=DISPLACEMENT
5.D-3
*BOUNDARY
PULL,2,2,7.
**
*EL PRINT,ELSET=CONTACT,POSITION=CENTROIDAL,FREQUENCY=200
PE
**
**
*OUTPUT,FIELD,FREQUENCY=1,VARIABLE=PRESELECT
*ELEMENT OUTPUT
S,PE
*NODE OUTPUT
U
*OUTPUT,HISTORY,FREQUENCY=1
*NODE OUTPUT,NSET=PULL
RF,U2
*NODE FILE,FREQUENCY=1,NSET=PULL
RF
*END STEP
```

Appendix B

Input File for the simulation of wire drawing using UEL with linear pressure

```
*HEADING
DRAWING
*RESTART,WRITE,FREQ=100
*NODE
1,0.,-3.05
51,1.225,-3.05
55,1.26,-3.05
5501,0,-0.05
5551,1.225,-0.05
5555,1.26,-0.05
****
****
*NGEN,NSET=BOTTOM1
1,51,1
*NGEN,NSET=BOTTOM2
51,55,1
*NGEN,NSET=TOP1
5501,5551,1
*NGEN,NSET=TOP2
5551,5555,1
*NFIL
BOTTOM1,TOP1,100,55
*NFIL
```

```
BOTTOM2,TOP2,100,55
*NSET,NSET=TOP
TOP1,TOP2
*NSET,NSET=LEFT,GENERATE
1,5501,55
*NSET,NSET=RIGHT1,GENERATE
51,5551,55
*NSET,NSET=RIGHT2,GENERATE
55,5555,55
*NSET,NSET=BOTTOM
BOTTOM1,BOTTOM2
**
**
*NODE
100000,0.,0.0
**
**
*NODE,NSET=PULL
200000,0.,0.
**
**
*USER ELEMENT, NODES=9, TYPE=U1, PROPERTIES=4, COORDINATES=3, VARIABLES=135, UNSYMM
1, 2, 4
5, 1, 2
**
**
*ELEMENT, TYPE=U1
 1, 1,3,113,111, 2,58,112,56, 57
**
*ELGEN,ELSET=ALLE
1,27,2,1,50,110,27
**
*UEL PROPERTY,ELSET=ALLE
216000.,0.295, 1., 5.
***
***ABAQUS ELEMENTS
***
**ELEMENT, TYPE=CPE8, ELSET=ABAQUS1
** 10001, 1,3,113,111, 2,58,112,56
**ELGEN,ELSET=ABAQUS
** 10001,27,2,1,50,110,27
***
```

```
**SOLID SECTION, ELSET=ABAQUS, MATERIAL=ZERO
**MATERIAL, NAME=ZERO
**USER MATERIAL, CONSTANTS=1
** 0.
**DEPVAR
** 1
**
***  For UVARM:
**USER OUTPUT VARIABLES
** 20
**
***ELSET,ELSET=OVERLAP,GENERATE
** 26,1349,27
** 27,1350,27
***UEL PROPERTY, ELSET=OVERLAP
** 107000.,0.33, 1., 5.
***
*SURFACE,TYPE=SEGMENTS,NAME=RIGID,FILLET RADIUS=0.055
START ,1.26 ,1.4715030
LINE ,1.181985 ,0.9715030
LINE ,1.181985 ,0.4987090
LINE ,1.26 ,0.0000000
LINE ,1.26 ,-3.0500000
*RIGID BODY,ANALYTICAL SURFACE=RIGID,REF NODE=100000
**
*SURFACE,NAME=OUT,TYPE=NODE
RIGHT2
**
*CONTACT PAIR,INTERACTION=ONE,TYPE=NODE TO SURFACE
OUT,RIGID
*SURFACE INTERACTION,NAME=ONE
**
*BOUNDARY
LEFT,1,1
100000,1,3
200000,1,1
200000,3,3
**
*EQUATION
2
TOP,2,1.,200000,2,-1.
**
```

```
**
** Tolerance parameters
*PARAMETER
 RTOL=5.E-4
 UTOL=1.E-3
 MTOL=5.E-4
 BTOL=1.E-3
**
**
*STEP,INC=1000,NLGEOM, UNSYMM=YES
*STATIC
0.001,0.05,,0.001
*CONTROLS, PARAMETERS=FIELD, FIELD=DISPLACEMENT
 <RTOL>, <UTOL>
*CONTROLS, PARAMETERS=FIELD, FIELD=ROTATION
 <MTOL>, <BTOL>
*BOUNDARY
PULL,2,2,0.05
**
*SOLVER CONTROLS, CONSTRAINT OPTIMIZATION
**
*PRINT, SOLVE=YES
**
*EL PRINT
*NODE PRINT
U1, U2, UR1, RF1, RF2, RM1
*EL PRINT, ELSET=ABAQUS
UVARM1, UVARM2, UVARM3, UVARM4
*OUTPUT, FIELD,FREQUENCY=1, VARIABLE=ALL
*END STEP
**
**
*STEP,INC=100000,NLGEOM, UNSYMM=YES
*STATIC
0.001,1.,,0.001
*CONTROLS, PARAMETERS=FIELD, FIELD=DISPLACEMENT
 <RTOL>, <UTOL>
*CONTROLS, PARAMETERS=FIELD, FIELD=ROTATION
 <MTOL>, <BTOL>
*BOUNDARY
PULL,2,2,1.
**
```

```
*SOLVER CONTROLS, CONSTRAINT OPTIMIZATION
**
*PRINT, SOLVE=YES
**
*EL PRINT
*NODE PRINT
U1, U2, UR1, RF1, RF2, RM1
*EL PRINT, ELSET=ABAQUS
UVARM1, UVARM2, UVARM3, UVARM4
*OUTPUT, FIELD,FREQUENCY=1, VARIABLE=AL
```

Appendix C

UEL subroutine for a mixed 9 node Lagrangian Element

```
      SUBROUTINE UEL(RHS,AMATRX,SVARS,ENERGY,NDOFEL,NRHS,NSVARS,PROPS,
     +NPROPS,COORDSS,MCRD,NNODE,U,DU,V,ACC,JTYPE,TIMEE,DTIME,KSTEP,KINC,
     +JELEM,PARAMS,NDLOAD,JDLTYP,ADLMAG,PREDEF,NPREDF,LFLAGS,MLVARX,
     +DDLMAG,MDLOAD,PNEWDT,JPROPS,NJPROP,PERIOD)
C
      INCLUDE 'ABA_PARAM.INC'
      CHARACTER*80 CMNAME
C
      DIMENSION RHS(MLVARX,NRHS),AMATRX(NDOFEL,NDOFEL),PROPS(NPROPS),
     + SVARS(NSVARS),ENERGY(8),COORDSS(MCRD,NNODE),U(NDOFEL),
     + DU(MLVARX,NRHS),V(NDOFEL),ACC(NDOFEL),TIMEE(2),PARAMS(3),
     + JDLTYP(MDLOAD,NRHS),ADLMAG(MDLOAD,NRHS),DDLMAG(MDLOAD,NRHS),
     + PREDEF(2,NPREDF,NNODE),LFLAGS(5),JPROPS(NJPROP)
C
! Note that for this element we have :
!       NNODE=9, NDOF=3, NGAUS=3
C       NDOFEL=NNODE*NDOF=4*3+5*2=22
C       NSVARS=NGAUS*NGAUS*13=3*3*15=99
C
C       NPROPS=4
C       PROPS = (E, ANU, SIG0, EXPO)
C
C       IAUX=(NPT-1)*NSVARS/(NGAUS*NGAUS)=(NPT-1)*15
C       SVARS(IAUX+1) = DFGRD1(1,1)
C       SVARS(IAUX+2) = DFGRD1(1,2)
C       SVARS(IAUX+3) = DFGRD1(2,1)
C       SVARS(IAUX+4) = DFGRD1(2,2)
C       SVARS(IAUX+5) = SDEV_11
C       SVARS(IAUX+6) = SDEV_22
C       SVARS(IAUX+7) = SDEV_33
C       SVARS(IAUX+8) = SDEV_12
C       SVARS(IAUX+9) = STATEV(1)   —— EBAR
```

```
C          SVARS(IAUX+10)= STATEV(2)  ---- YFLAG
C          SVARS(IAUX+11)= PRESS
C          SVARS(IAUX+12)= STRESS_11
C          SVARS(IAUX+13)= STRESS_22
C          SVARS(IAUX+14)= STRESS_33
C          SVARS(IAUX+15)= STRESS_12
C
C
C
!
! If we have more than 2 DOF for nodes,
! Abaqus assumes that it is a 3D analysis and setS MMCRD = 3.
! Then we introduced MCRD=2 and define COORDS based on 2D coordinate system.
!
! We used TIMEE instead of TIME. Since sometimes TIME is reserved by Fortran.
C
      DIMENSION POSGP(3),WEIGP(3),SNINE(9),DNINE(2,9),SFOUR(4),
     + DFOUR(2,4),ANUMATX(2,22),BDEV(4,22),BDEVTRAN(22,4),
     + FBAR(2,2),XJACM(2,2),XJACM0(2,2),CARTD9(2,9),CARTD90(2,9),
     + CARTD90T(9,2),COORDS0(2,9),COORDS(2,9),RHSS(22)
      DIMENSION ANPMATX(22),BV(22),DUU(22)
C
C*** Dimensions for KUMAT
C*** 2-problems with NDI=3, NSHR=1, NTENS=4, NSTATV=2, NPROPSU=4
      DIMENSION SDEV(4),STRESS(4),STATEV(2),SEL(4),
     + DDSDDE(4,4),PROPSU(4),COORDSU(2),
     + DROT(3,3),DFGRD0(3,3),DFGRD1(3,3)
C*** THE FIRST INDEX IN THE DIMENSION OF ZXEN SHOULD BE >=
C     THE TOTAL NUMBER OF USER ELEMENTS
      DIMENSION ZXEN(2000,9,20)
      COMMON/KNICK/ZXEN
C
C
      IWR = 0
      IF (JELEM == 1) IWR = 00
C
      IOUT = 7      ! IOUT=7 WRITES ON THE .msg FILE
C*** FOR NEW FILES (in Abaqus/Standard) USE 14<IOUT<19 OR IOUT>100
C     SEE ABAQUS 6.13 USER'S GUIDE SECTION 3.7.1


C-----------------------------------------------------------------
C*****************************************************************
      LAPPLY_FORCE = 0
C*** BOOLEAN VARIABLE TO AUTOMATICALLY CHANGE THE RHS AND AMATRX SIGNS
C     IN THE CASE WHERE WE APPLY CONCENTRATED FORCES ON THIS ELEMENT
C*****************************************************************
C-----------------------------------------------------------------


C
C*** PLANE-STRAIN AND AXISYMMETRIC PROBLEMS
      NDI = 3
      NSHR = 1
      NTENS = NDI + NSHR
C*** STATEV(I): EPBAR, YFLAG
```

```
89          NSTATV = 2
90    C*** PROPS(I): E, ANU, SIG0, EXPO
91          NPROPSU = NPROPS
92    C*** ABAQUS passes in MCRD=3 because of the dofs used in the element
93          MMCRD = 2
94    C
95    C
96          IF (IWR /= 0) THEN
97            WRITE(IOUT,*)
98            WRITE(IOUT,*)
99            WRITE(IOUT,*) ' UEL STARTS ----------------------------------------'
100           WRITE(IOUT,*)
101           WRITE(IOUT,*) '-------------------------------------------------'
102           WRITE(IOUT,*) 'STEP TIME, DTIME, TOTAL TIME'
103           WRITE(IOUT,1001) TIMEE(1),DTIME,TIMEE(2)
104           WRITE(IOUT,*) 'JELEM, KINC, KSTEP'
105           WRITE(IOUT,1002) JELEM, KINC, KSTEP
106           WRITE(IOUT,*) 'NDI, NSHR, NTENS'
107           WRITE(IOUT,1002) NDI,NSHR,NTENS
108           WRITE(IOUT,*) 'MLVARX, NRHS, NNODE, NDOFEL, NSVARS, MCRD,NPROPS'
109           WRITE(IOUT,1002) MLVARX,NRHS,NNODE,NDOFEL,NSVARS,MCRD,NPROPS
110           WRITE(IOUT,*) ' MMCRD'
111           WRITE(IOUT,1002) MMCRD
112           WRITE(IOUT,*) "U"
113           DO INODE=1,4
114             IAUX = (INODE-1)*3
115             WRITE(IOUT,1001) (U(J),J=IAUX+1,IAUX+3)
116           END DO
117           DO INODE=5,9
118             IAUX = (INODE-5)*2
119             WRITE(IOUT,1001) (U(J),J=IAUX+1,IAUX+2)
120           END DO
121           WRITE(IOUT,*) 'DU'
122           DO INODE=1,4
123             IAUX = (INODE-1)*3
124             WRITE(IOUT,1001) (DU(J,1),J=IAUX+1,IAUX+3)
125           END DO
126           DO INODE=5,9
127             IAUX = (INODE-5)*2
128             WRITE(IOUT,1001) (DU(J,1),J=IAUX+1,IAUX+2)
129           END DO
130         END IF
131   C
132   C*** Copy material properties to be used in UMAT (E, ANU, SIG0, EXPO)
133         PROPSU = PROPS ! PROPSU(I) = PROPS(I)
134   C
135         NGAUS=3
136         POSGP = [ -DSQRT(0.6D0), DSQRT(0.0D0), DSQRT(0.6D0)]
137         WEIGP = [ 5.D0/9.D0    , 8.D0/9.D0, 5.D0/9.D0    ]
138   C
139         IF (IWR /= 0) THEN
140           WRITE(IOUT,*) 'PROPS: E, ANU, SIG0, EXPO'
141           WRITE(IOUT,1001) (PROPS(I),I=1,NPROPS)
142           WRITE(IOUT,*) 'PROPSU'
143           WRITE(IOUT,1001) (PROPSU(I),I=1,NPROPSU)
144           WRITE(IOUT,*) 'POSGP'
```

```
145           WRITE(IOUT,1001) (POSGP(I),I=1,NGAUS)
146           WRITE(IOUT,*) 'WEIGP'
147           WRITE(IOUT,1001) (WEIGP(I),I=1,NGAUS)
148         END IF
149 C
150         E    = PROPS(1)
151         ANU  = PROPS(2)
152         SIG0 = PROPS(3)
153         EXPO = PROPS(4)
154         E0   = SIG0/E
155 C*** NODAL COORDINATES IN THE UNDEFORMED (COORDS0) AND
156 C*** DEFORMED END_OF_INCREMENT CONFIGURATION (COORDS)
157         COORDS0(1:MMCRD,1:NNODE) = COORDSS(1:MMCRD,1:NNODE)
158         DO INODE=1,4
159           I = (INODE-1)*3 + 1
160           COORDS(1,INODE) = COORDS0(1,INODE) + U(I)
161           COORDS(2,INODE) = COORDS0(2,INODE) + U(I+1)
162         END DO
163         DO INODE=5,9
164           I = 13 + (INODE-5)*2
165           COORDS(1,INODE) = COORDS0(1,INODE) + U(I)
166           COORDS(2,INODE) = COORDS0(2,INODE) + U(I+1)
167         END DO
168         IF (IWR.NE.0) THEN
169             WRITE(IOUT,*) ' COORDS'
170             WRITE(IOUT,1001) ((COORDS(J,I),J=1,2),I=1,9)
171         END IF
172 C
173 C
174         RHSS = 0.D0    ! RHSS(I) = 0.
175         RHS = 0.D0     ! RHS(I) = 0.
176         AMATRX = 0.D0 ! AMATRX(I,J) = 0.
177 C
178 C
179 C ---- Initialize the state variables at the beginning
180         IF (KSTEP <= 1 .AND. KINC <= 1) THEN
181           SVARS = 0.D0  !  SVARS(I)=0
182           NPT = 0
183           DO IGAUS=1,NGAUS
184           DO JGAUS=1,NGAUS
185             NPT = NPT + 1
186             IAUX = (NPT-1)*NSVARS/(NGAUS*NGAUS)
187             SVARS(IAUX+1) = 1.D0 ! DFGRD0(1,1)
188             SVARS(IAUX+4) = 1.D0 ! DFGRD0(2,2)
189           END DO
190           END DO
191         END IF
192 C
193 C
194 C*** DO LOOP OVER GAUSS POINTS
195         IWR0 = IWR
196         NPT = 0
197         DO IGAUS=1,NGAUS
198         DO JGAUS=1,NGAUS
199 C
200           NPT = NPT + 1
```

```
C          IWR = 0
C          IF (IWR0.NE.0.AND.NPT.EQ.9) IWR = 01
C
          XI = POSGP(JGAUS)
          ETA = POSGP(IGAUS)
C
C*** CALCULATE SHAPE FUNCTIONS
          CALL KSHAPE4(XI,ETA,SFOUR,DFOUR)
          CALL KSHAPE9(XI,ETA,SNINE,DNINE)
C
C*** CALCULATE NU-MATRIX AND NP-MATRIX
          CALL KNUMATRIX(ANUMATX,SNINE) ! Calculates the Nu matrix
          CALL KNPMATRIX(ANPMATX,SFOUR) ! Calculates the Np matrix
C
          IF (IWR /= 0) THEN
            WRITE(IOUT,*)
            WRITE(IOUT,*)
            WRITE(IOUT,*)
            WRITE(IOUT,*) 'LOOP OVER GAUSS POINTS STARTS'
            WRITE(IOUT,*) 'NPT '
            WRITE(IOUT,1002) NPT
            WRITE(IOUT,*) 'XI, ETA '
            WRITE(IOUT,1001) XI, ETA
            WRITE(IOUT,*) 'Sum(SNINE) '
            WRITE(IOUT,1001) SUM(SNINE)
            WRITE(IOUT,*) 'Sum(DNINE(1,:)) '
            WRITE(IOUT,1001) SUM(DNINE(1,:))
            WRITE(IOUT,*) 'Sum(DNINE(2,:))'
            WRITE(IOUT,1001) SUM(DNINE(2,:))
            WRITE(IOUT,*) 'SNINE'
            WRITE(IOUT,1001) SNINE
            WRITE(IOUT,*) 'DNINE'
            DO I=1,MMCRD
              WRITE(IOUT,1001) (DNINE(I,J),J=1,NNODE)
            ENDDO
            WRITE(IOUT,*) 'ANUMATX'
            DO I=1,MMCRD
              WRITE(IOUT,1001) (ANUMATX(I,J),J=1,12)
            ENDDO
            WRITE(IOUT,*)
            DO I=1,MMCRD
              WRITE(IOUT,1001) (ANUMATX(I,J),J=13,22)
            ENDDO
            WRITE(IOUT,*) 'ANPMATX'
            WRITE(IOUT,1001) (ANPMATX(I),I=1,12)
            WRITE(IOUT,*)
            WRITE(IOUT,1001) (ANPMATX(I),I=13,22)
            WRITE(IOUT,*)
          END IF
C
C*** CALCULATE JACOBIAN MATRIX (XJACM) AND ITS DETERMINANT (DJACB)
          CALL KJACOB2D(XJACM,DJACB,DNINE,COORDS,NNODE,
     +                  JELEM,NPT,IOUT)
C
          IF (IWR /= 0) THEN
            WRITE(IOUT,*) 'DJACB'
```

```
257             WRITE(IOUT,1001) DJACB
258             WRITE(IOUT,*) 'XJACM'
259             DO I=1,2
260               WRITE(IOUT,1001) (XJACM(I,J),J=1,2)
261             ENDDO
262           END IF
263 C
264 C*** CALCULATE CARTESIAN DERIVATIVES OF SHAPE FUNCTIONS
265           CALL KCARTD(CARTD9,XJACM,DNINE,NNODE)
266 C
267           IF (IWR /= 0) THEN
268             WRITE(IOUT,*) 'CARTD9(1,:)'
269             WRITE(IOUT,1001) CARTD9(1,:)
270             WRITE(IOUT,*) 'CARTD9(2,:)'
271             WRITE(IOUT,1001) CARTD9(2,:)
272             WRITE(IOUT,*) 'Sum(CARTD9(1,:))'
273             WRITE(IOUT,1001) SUM(CARTD9(1,:))
274             WRITE(IOUT,*) 'Sum(CARTD9(2,:))'
275             WRITE(IOUT,1001) SUM(CARTD9(2,:))
276           END IF
277 C
278 C*** CALCULATE BDEV-MATRIX AND BV-MATRIX
279           CALL KBMATRIX(BDEV,BV,DJACB,SNINE,DNINE,COORDS,NNODE,JELEM,
280      +                  NPT,IOUT)
281           IF (IWR.NE.0) THEN
282             WRITE(IOUT,*) '[BDEV]'
283             DO I=1,NTENS
284               WRITE(IOUT,1001) (BDEV(I,J),J=1,NDOFEL)
285             END DO
286             WRITE(IOUT,*) '[BV]'
287             WRITE(IOUT,1001) (BV(I),I=1,NDOFEL)
288           END IF
289 C
290 C*** READ DFRGR0, STRESSES, AND STATE VARIABLES AT THE START OF THE INCREMENT
291           IAUX=(NPT-1)*NSVARS/(NGAUS*NGAUS)
292           DFGRD0 = 0.D0 ! DFGRD0(I,J) = 0.
293           DFGRD0(1,1) = SVARS(IAUX+1)
294           DFGRD0(1,2) = SVARS(IAUX+2)
295           DFGRD0(2,1) = SVARS(IAUX+3)
296           DFGRD0(2,2) = SVARS(IAUX+4)
297           DFGRD0(3,3) = 1.D0
298           DO I=1,NTENS  !NTENS=4
299             SDEV(I) = SVARS(IAUX+4+I)
300           ENDDO
301           STATEV(1)  = SVARS(IAUX+9)     ! EBAR
302           STATEV(2)  = SVARS(IAUX+10)    ! YFLAG
303           PRESSN     = SVARS(IAUX+11)
304           DO I=1,NTENS  !NTENS=4
305             STRESS(I) = SVARS(IAUX+11+I)
306           ENDDO
307 C
308           IF (IWR /= 0) THEN
309             WRITE(IOUT,*) ' STATEV'
310             WRITE(IOUT,1001) STATEV(1),STATEV(2)
311             WRITE(IOUT,*) ' PRESSN'
312             WRITE(IOUT,1001) PRESSN
```

```
313           END IF
314   C
315   C*** CALCULATE DFGRD1 AT THE END OF THE INCREMENT
316   !--- First calculate Cartesian derivative of the shape function in the
317   !     undeformed configuration, CARTD90, then obtain deformation gradient, FBAR
318           CALL KJACOB2D(XJACM0,DJACB0,DNINE,COORDS0,NNODE,
319         +                       JELEM,NPT,IOUT)
320           CALL KCARTD(CARTD90,XJACM0,DNINE,NNODE)
321   C
322           CARTD90T = Transpose(CARTD90)
323           FBAR = MATMUL(COORDS,CARTD90T)
324           DFGRD1 = 0.D0     ! DFGRD1(I,J) = 0.
325           DFGRD1(1:2,1:2) = FBAR(1:2,1:2)
326           DFGRD1(3,3) = 1.D0
327           CALL KDET3X3(DFGRD1,DET1)
328           IF (IWR.NE.0) THEN
329             WRITE(IOUT,*) ' DFGRD1'
330             DO I=1,3
331               WRITE(IOUT,1001) (DFGRD1(I,J),J=1,3)
332             END DO
333             WRITE(IOUT,*) ' DET '
334             WRITE(IOUT,1001) DET1
335           END IF
336   C
337   C
338   C*** CALCULATE CARTESIAN COORDINATES OF THE GAUSS POINT (COORDSU)
339           COORDSU = MATMUL(COORDS0,SNINE)
340   C
341   C*** FIND PRESS, DPRESS, DIVDU
342   C
343           DUU(1:NDOFEL) = DU(1:NDOFEL,1)
344   C
345           DPRESS = DOT_PRODUCT(ANPMATX,DUU)
346           PRESS = PRESSN + DPRESS
347           DIVDU = DOT_PRODUCT(BV,DUU)
348           IF (IWR /= 0) THEN
349             WRITE(IOUT,*) ' PRESSN, DPRESS, PRESS'
350             WRITE(IOUT,1001) PRESSN, DPRESS, PRESS
351             WRITE(IOUT,*) ' DIVDU'
352             WRITE(IOUT,1001) DIVDU
353           END IF
354   C
355   C*** CALL 'UMAT' FOR MATERIAL CALCULATIONS
356           SIGY = 0.D0
357           NOEL = JELEM
358           CALL KUMAT(SDEV,STATEV,DDSDDE,TIMEE,DTIME,CMNAME,
359         +     NDI,NSHR,NTENS,NSTATV,PROPSU,NPROPS,COORDSU,DROT,PNEWDT,
360         +     DFGRD0,DFGRD1,NOEL,NPT,KSTEP,KINC,IOUT,IWR,SIGY,DEKK)
361           IF (PNEWDT < 1.D0) RETURN
362   C
363   C
364   C*** CALCULATE RESIDUAL, RHS
365           IF (IWR /= 0) THEN
366             WRITE(IOUT,*) 'SDEV'
367             WRITE(IOUT,1001) (SDEV(I),I=1,NTENS)
368             WRITE(IOUT,*) ' STATEV(1), STATEV(2)'
```

```
369               WRITE(IOUT,1001) STATEV(1), STATEV(2)
370               WRITE(IOUT,*) ' DDSDDE'
371               DO I=1,NTENS
372                 WRITE(IOUT,1001) (DDSDDE(I,J),J=1,NTENS)
373               END DO
374             END IF
375 C
376             BDEVTRAN = Transpose(BDEV)
377 C
378 !------------------------ CALCULATE RESIDUAL VECTOR RHS
379             DVOLU = DJACB * WEIGP(IGAUS) * WEIGP(JGAUS)
380 C
381             CALL KRHSS(RHSS,SDEV,BDEVTRAN,BV,DVOLU,DPRESS,
382      +                 PRESS,ANPMATX,E,ANU,DEKK,I_APPLY_FORCE)
383 C
384             IF (IWR /= 0) THEN
385               WRITE(IOUT,*) 'value of RHSS up to this NPT'
386               WRITE(IOUT,1001) RHSS
387           END IF
388
389 C
390 C
391 !------------------------ CALCULATE JACOBIAN MATRIX, AMATRX
392 C
393             CALL KAMATRIX(AMATRX,DDSDDE,BDEV,BDEVTRAN,BV,ANPMATX,DVOLU,
394      + E,ANU,NDOFEL,I_APPLY_FORCE)
395 C
396             STRESS(1:NTENS) = SDEV(1:NTENS)
397             STRESS(1:NDI) = SDEV(1:NDI) + PRESS
398 C
399 C*** UPDATE STATE VARIABLES
400             IAUX = (NPT-1)*NSVARS/(NGAUS*NGAUS)
401             SVARS(IAUX+1)  = DFGRD1(1,1)
402             SVARS(IAUX+2)  = DFGRD1(1,2)
403             SVARS(IAUX+3)  = DFGRD1(2,1)
404             SVARS(IAUX+4)  = DFGRD1(2,2)
405 C
406             SVARS(IAUX+5)  = SDEV(1)
407             SVARS(IAUX+6)  = SDEV(2)
408             SVARS(IAUX+7)  = SDEV(3)
409             SVARS(IAUX+8)  = SDEV(4)
410 C
411             SVARS(IAUX+9)  = STATEV(1)
412             SVARS(IAUX+10) = STATEV(2)
413             SVARS(IAUX+11) = PRESS
414 C
415             SVARS(IAUX+12)  = STRESS(1)
416             SVARS(IAUX+13)  = STRESS(2)
417             SVARS(IAUX+14)  = STRESS(3)
418             SVARS(IAUX+15)  = STRESS(4)
419 C
420 C
421 C
422 C
423 C
424 C*** STORE VARIABLES FOR CONTOUR PLOTTING
```

```
425 C     IN ABAQUS INPUT FILE INCLUDE FOR THE ABAQUS ZERO STIFFNESS ELEMENT
426 C     WITH UMAT:
427 C     *USER OUTPUT VARIABLES
428 C       20
429 C     WHERE 20 IS THE MAX NUMBER OF VARIABLES THAT CAN BE STORED
430 C     (INCREASE IF NEEDED)
431 C     SEE ALSO USER SUBROUTINE UVARM AT THE END OF THIS FILE
432 C
433
434           CALL KINVAR(STRESS,PRESS,Q,NDI,NSHR,NTENS)
435
436           ZXEN(JELEM,NPT,1:NTENS) = STRESS(1:NTENS)
437           ZXEN(JELEM,NPT,5:8) = SDEV(1:NTENS)
438           ZXEN(JELEM,NPT,9) = PRESS
439           ZXEN(JELEM,NPT,10)=Q
440           ZXEN(JELEM,NPT,11) = STATEV(1)
441           ZXEN(JELEM,NPT,12) = STATEV(2)
442           ZXEN(JELEM,NPT,13) = SIGY
443 C
444 C
445         ENDDO
446         ENDDO
447 C*** END OF LOOP OVER GAUSS POINTS
448 C
449 C
450 C
451 C
452 C
453         IWR = IWR0
454         RHS(1:NDOFEL,1) = RHSS(1:NDOFEL)
455         IF (IWR /= 0) THEN
456           WRITE(IOUT,*)
457           WRITE(IOUT,*)
458           WRITE(IOUT,*)
459           WRITE(IOUT,*) 'LOOP OVER GAUSS POINTS FINISHED'
460           WRITE(IOUT,*) 'RHS '
461           WRITE(IOUT,1001) RHS
462           WRITE(IOUT,*)
463           WRITE(IOUT,*) 'AMATRX(1:8,:) '
464           DO I=1,8
465             WRITE(IOUT,1001) (AMATRX(I,J),J=1,NDOFEL)
466           END DO
467         END IF
468 C
469 C
470  1001 FORMAT(1P8E13.5)
471  1002 FORMAT(10I5)
472 C
473         END
474 C
475 C**********************************************************************
476 C
477         SUBROUTINE KSHAPE4(XI,ETA,SFOUR,DFOUR)
478 C
479 C*** CALCULATES SHAPE FCNS AND THEIR DERIVATIVES FOR 4-NODE 2D ELEMENTS
480 C
```

```
481         IMPLICIT DOUBLE PRECISION(A-H,O-Z)
482         DIMENSION SFOUR(4),DFOUR(2,4)
483   C
484         SFOUR(1) = 0.25D0*(1.D0 - XI)*(1.D0 - ETA)
485         SFOUR(2) = 0.25D0*(1.D0 + XI)*(1.D0 - ETA)
486         SFOUR(3) = 0.25D0*(1.D0 + XI)*(1.D0 + ETA)
487         SFOUR(4) = 0.25D0*(1.D0 - XI)*(1.D0 + ETA)
488   C
489   C*** SHAPE FUNCTION DERIVATIVES
490   C
491         DFOUR(1,1) = -0.25D0*(1.D0 - ETA)
492         DFOUR(1,2) =  0.25D0*(1.D0 - ETA)
493         DFOUR(1,3) =  0.25D0*(1.D0 + ETA)
494         DFOUR(1,4) = -0.25D0*(1.D0 + ETA)
495   C
496         DFOUR(2,1) = -0.25D0*(1.D0 - XI)
497         DFOUR(2,2) = -0.25D0*(1.D0 + XI)
498         DFOUR(2,3) =  0.25D0*(1.D0 + XI)
499         DFOUR(2,4) =  0.25D0*(1.D0 - XI)
500   C
501         END
502   C
503   C*********************************************************************
504   C
505         SUBROUTINE KSHAPE9(XI,ETA,SNINE,DNINE)
506   C
507   C*** CALCULATES SHAPE FCNS AND THEIR DERIVATIVES FOR 9-NODE 2D ELEMENTS
508   C
509         IMPLICIT DOUBLE PRECISION(A-H,O-Z)
510         DIMENSION SNINE(9),DNINE(2,9)
511   C
512         XI2 = XI*XI
513         ETA2 = ETA*ETA
514         SNINE(1) = 0.25D0*(XI2 - XI)*(ETA2 - ETA)
515         SNINE(2) = 0.25D0*(XI2 + XI)*(ETA2 - ETA)
516         SNINE(3) = 0.25D0*(XI2 + XI)*(ETA2 + ETA)
517         SNINE(4) = 0.25D0*(XI2 - XI)*(ETA2 + ETA)
518         SNINE(5) = 0.50D0*(1.D0 - XI2)*(ETA2 - ETA)
519         SNINE(6) = 0.50D0*(XI2 + XI)*(1.D0 - ETA2)
520         SNINE(7) = 0.50D0*(1.D0 - XI2)*(ETA2 + ETA)
521         SNINE(8) = 0.50D0*(XI2 - XI)*(1.D0 - ETA2)
522         SNINE(9) = (1.D0 - XI2)*(1.D0 - ETA2)
523   C
524   C***** SHAPE FUNCTION DERIVATIVES
525   C
526         DNINE(1,1) =  0.25D0*(2.D0*XI - 1.D0)*(ETA2 - ETA)
527         DNINE(1,2) =  0.25D0*(2.D0*XI + 1.D0)*(ETA2 - ETA)
528         DNINE(1,3) =  0.25D0*(2.D0*XI + 1.D0)*(ETA2 + ETA)
529         DNINE(1,4) =  0.25D0*(2.D0*XI - 1.D0)*(ETA2 + ETA)
530         DNINE(1,5) =  -XI*(ETA2-ETA)
531         DNINE(1,6) =  0.50D0*(2.D0*XI + 1.D0)*(1.D0 - ETA2)
532         DNINE(1,7) =  -XI*(ETA2 + ETA)
533         DNINE(1,8) =  0.50D0*(2.D0*XI - 1.D0)*(1.D0 - ETA2)
534         DNINE(1,9) =  -2.D0*XI*(1.D0 - ETA2)
535   C
536         DNINE(2,1) =  0.25D0*(XI2 - XI)*(2.D0*ETA - 1.D0)
```

```
537         DNINE(2,2) =  0.25D0*(XI2 + XI)*(2.D0*ETA - 1.D0)
538         DNINE(2,3) =  0.25D0*(XI2 + XI)*(2.D0*ETA + 1.D0)
539         DNINE(2,4) =  0.25D0*(XI2 - XI)*(2.D0*ETA + 1.D0)
540         DNINE(2,5) =  0.50D0*(1.D0 - XI2)*(2.D0*ETA - 1.D0)
541         DNINE(2,6) = -(XI2 + XI)*ETA
542         DNINE(2,7) =  0.50D0*(1.D0 - XI2)*(2.D0*ETA + 1.D0)
543         DNINE(2,8) = -(XI2 - XI)*ETA
544         DNINE(2,9) = -(1.D0 - XI2)*2.D0*ETA
545 C
546         END
547 C
548 C*********************************************************************
549 C
550         SUBROUTINE KINV2X2(A,AINV)
551 C
552 C*** INVERTS 2X2 MATRIX
553 C
554         IMPLICIT REAL*8(A-H,O-Z)
555 C
556         DIMENSION A(2,2),AINV(2,2)
557 C
558         DET   = A(1,1)*A(2,2) - A(1,2)*A(2,1)
559         ANORM = DSQRT(   A(1,1)*A(1,1) + A(1,2)*A(1,2)
560        +               + A(2,1)*A(2,1) + A(2,2)*A(2,2) )
561         TOL = ANORM*1.D-8
562         IF (DET <= TOL) THEN
563           WRITE(*,*) 'TRYING TO INVERT SINGULAR 2X2 MATRIX'
564           WRITE(*,*) 'PROGRAM STOPS.'
565           CALL XIT
566         END IF
567         AINV(1,1) =  A(2,2)/DET
568         AINV(2,2) =  A(1,1)/DET
569         AINV(1,2) = -A(1,2)/DET
570         AINV(2,1) = -A(2,1)/DET
571 C
572         END
573 C
574 C*********************************************************************
575 C
576         SUBROUTINE KINV3X3(A,AINV)
577 C
578         IMPLICIT REAL*8(A-H,O-Z)
579 C
580         DIMENSION A(3,3),AINV(3,3)
581 C
582         DET =  A(1,1)*(A(2,2)*A(3,3) - A(3,2)*A(2,3))
583        +     - A(1,2)*(A(2,1)*A(3,3) - A(3,1)*A(2,3))
584        +     + A(1,3)*(A(2,1)*A(3,2) - A(3,1)*A(2,2))
585         ANORM = DSQRT( A(1,1)*A(1,1) + A(1,2)*A(1,2) + A(1,3)*A(1,3)
586        +             + A(2,1)*A(2,1) + A(2,2)*A(2,2) + A(2,3)*A(2,3)
587        +             + A(3,1)*A(3,1) + A(3,2)*A(3,2) + A(3,3)*A(3,3) )
588         TOL = ANORM*1.D-8
589 C
590         IF (DET <= TOL) THEN ! Mohsen
591           WRITE(*,*) 'TRYING TO INVERT SINGULAR 3X3 MATRIX'
592           WRITE(*,*) 'PROGRAM STOPS.'
```

```
593           CALL XIT
594         END IF
595 C
596         AINV(1,1) =  (A(2,2)*A(3,3) - A(2,3)*A(3,2))/DET
597         AINV(1,2) = -(A(1,2)*A(3,3) - A(3,2)*A(1,3))/DET
598         AINV(1,3) =  (A(1,2)*A(2,3) - A(2,2)*A(1,3))/DET
599         AINV(2,1) = -(A(2,1)*A(3,3) - A(3,1)*A(2,3))/DET
600         AINV(2,2) =  (A(1,1)*A(3,3) - A(3,1)*A(1,3))/DET
601         AINV(2,3) = -(A(1,1)*A(2,3) - A(2,1)*A(1,3))/DET
602         AINV(3,1) =  (A(2,1)*A(3,2) - A(3,1)*A(2,2))/DET
603         AINV(3,2) = -(A(1,1)*A(3,2) - A(3,1)*A(1,2))/DET
604         AINV(3,3) =  (A(1,1)*A(2,2) - A(2,1)*A(1,2))/DET
605 C
606         END
607 C
608 C*********************************************************************
609 C
610         SUBROUTINE KJACOB2D(XJACM,DJACB,DSHAPE,COORDS,NNODE,
611        +                    IELEM,KGAUS,IOUT)
612 C
613 C*** 2D ELEMENT (MMCRD=2) WITH NNODE NODES, EVALUATES:
614 C*** JACOBIAN MATRIX (XJACM) AND ITS DETERMINANT (DJACB)
615 C
616         IMPLICIT DOUBLE PRECISION(A-H,O-Z)
617 C
618         DIMENSION XJACM(2,2),DSHAPE(2,NNODE),COORDS(2,NNODE),
619        + COORDST(NNODE,2)
620 C
621 C*** CALCULATE JACOBIAN MATRIX (XJACM)
622
623         COORDST = Transpose(COORDS)
624         XJACM = MATMUL(DSHAPE,COORDST)
625 C
626 C*** CALCULATE DETERMINANT OF 2X2 JACOBIAN MATRIX (DJACB)
627         DJACB = XJACM(1,1)*XJACM(2,2) - XJACM(1,2)*XJACM(2,1)
628 C
629         IF (DJACB <= 0.D0) THEN
630           WRITE(IOUT,*) 'NON-POSITIVE JACOBIAN IN ELEMENT ',IELEM
631           WRITE(IOUT,*) 'GAUSS POINT ',KGAUS
632           WRITE(IOUT,*) 'NODE',          NNODE
633           WRITE(IOUT,*) 'PROGRAM STOPS.'
634           CALL XIT
635         END IF
636 C
637         END
638 C
639 C*********************************************************************
640 C
641         SUBROUTINE KCARTD(CARTD,XJACM,DSHAPE,MNODE)
642 C
643 C*** 2D ELEMENT (MMCRD=2) WITH NNODE NODES, EVALUATES:
644 C*** DERIVATIVES OF SHAPE FCNS WRT X AND Y (CARTD)
645 C
646         IMPLICIT DOUBLE PRECISION(A-H,O-Z)
647 C
648         DIMENSION CARTD(2,MNODE),XJACM(2,2),DSHAPE(2,MNODE),
```

```
649         + XJACI(2,2)
650   C
651   C*** CALCULATE THE INVERSE OF 2X2 JACOBIAN MATRIX (XJACI)
652         CALL KINV2X2(XJACM,XJACI)
653   C
654   C*** CALCULATE CARTESIAN DERIVATIVES OF SHAPE FCNS (CARTD)
655         CARTD = MATMUL(XJACI,DSHAPE)
656   C
657         END
658   C
659   C**********************************************************************
660   C
661         SUBROUTINE KNUMATRIX(ANUMATX,SNINE)
662   C
663   C*** CALCULATES MATRIX N_u
664   C
665         IMPLICIT DOUBLE PRECISION(A-H,O-Z)
666   C
667         DIMENSION ANUMATX(2,22),SNINE(9)
668   C
669         ANUMATX = 0.D0 ! ANUMATX(I,J) = 0.
670   C
671         DO INODE=1,4
672         DO IDOF=1,2
673            I = (INODE-1)*3 + IDOF
674            ANUMATX(IDOF,I) = SNINE(INODE)
675         ENDDO
676         ENDDO
677   C
678         DO INODE=5,9
679         DO IDOF=1,2
680            I = 12 + (INODE-5)*2 + IDOF
681            ANUMATX(IDOF,I) = SNINE(INODE)
682         ENDDO
683         ENDDO
684   C
685         END
686   C
687   C**********************************************************************
688   C
689         SUBROUTINE KNPMATRIX(ANPMATX,SFOUR)
690   C
691   C*** CALCULATES MATRIX N_p
692   C
693         IMPLICIT DOUBLE PRECISION(A-H,O-Z)
694   C
695         DIMENSION ANPMATX(22),SFOUR(4)
696   C
697         ANPMATX = 0.D0 ! ANPMATX(I,J) = 0.
698   C
699         DO INODE=1,4
700            I = INODE*3
701            ANPMATX(I) = SFOUR(INODE)
702         ENDDO
703   C
704         END
```

```
705 C
706 C**********************************************************************
707 C
708       SUBROUTINE KBMATRIX(BDEV,BV,DJACB,SNINE,DSHAPE,COORDS,NNODE,
709      + IELEM,KGAUS,IOUT)
710 C
711 C*** CALCULATES MATRIX B
712 C
713       IMPLICIT DOUBLE PRECISION(A-H,O-Z)
714 C
715       DIMENSION BDEV(4,22),BV(22),CARTD9(2,9),XJACM(2,2)
716       DIMENSION SNINE(NNODE),DSHAPE(2,NNODE),COORDS(2,NNODE)
717 C
718       CALL KJACOB2D(XJACM,DJACB,DSHAPE,COORDS,NNODE,IELEM,KGAUS,IOUT)
719       CALL KCARTD(CARTD9,XJACM,DSHAPE,NNODE)
720 C
721       BDEV = 0.D0 ! BDEV(I,J) = 0.
722       BV   = 0.D0 ! BV(I,J)   = 0.
723 C
724       DO INODE=1,4
725          I = (INODE-1)*3 + 1
726          BDEV(1,I)   =  (2.D0/3.D0)*CARTD9(1,INODE)
727          BDEV(1,I+1) = -(1.D0/3.D0)*CARTD9(2,INODE)
728          BDEV(2,I)   = -(1.D0/3.D0)*CARTD9(1,INODE)
729          BDEV(2,I+1) =  (2.D0/3.D0)*CARTD9(2,INODE)
730          BDEV(3,I)   = -(1.D0/3.D0)*CARTD9(1,INODE)
731          BDEV(3,I+1) = -(1.D0/3.D0)*CARTD9(2,INODE)
732          BDEV(4,I)   =  CARTD9(2,INODE)
733          BDEV(4,I+1) =  CARTD9(1,INODE)
734       ENDDO
735 C
736       DO INODE=5,9
737          I = 13 + (INODE-5)*2
738          BDEV(1,I)   =  (2.D0/3.D0)*CARTD9(1,INODE)
739          BDEV(1,I+1) = -(1.D0/3.D0)*CARTD9(2,INODE)
740          BDEV(2,I)   = -(1.D0/3.D0)*CARTD9(1,INODE)
741          BDEV(2,I+1) =  (2.D0/3.D0)*CARTD9(2,INODE)
742          BDEV(3,I)   = -(1.D0/3.D0)*CARTD9(1,INODE)
743          BDEV(3,I+1) = -(1.D0/3.D0)*CARTD9(2,INODE)
744          BDEV(4,I)   =  CARTD9(2,INODE)
745          BDEV(4,I+1) =  CARTD9(1,INODE)
746       ENDDO
747 C
748       DO INODE=1,4
749          I = (INODE-1)*3 + 1
750          BV(I)   = CARTD9(1,INODE)
751          BV(I+1) = CARTD9(2,INODE)
752       ENDDO
753 C
754       DO INODE=5,9
755          I = 13 + (INODE-5)*2
756          BV(I)   = CARTD9(1,INODE)
757          BV(I+1) = CARTD9(2,INODE)
758       ENDDO
759 C
760       END
```

```
C
C*****************************************************************************
C
      SUBROUTINE KDET3X3(A,DET)
C
      IMPLICIT REAL*8(A-H,O-Z)
C
      DIMENSION A(3,3)
C
      DET =  A(1,1)*(A(2,2)*A(3,3) - A(3,2)*A(2,3))
     +     - A(1,2)*(A(2,1)*A(3,3) - A(3,1)*A(2,3))
     +     + A(1,3)*(A(2,1)*A(3,2) - A(3,1)*A(2,2))
C
      END
C
C*****************************************************************************
C
      SUBROUTINE KRHSS(RHSS,SDEV,BDEVTRAN,BV,DVOLU,DPRESS,
     + PRESS,ANPMATX,E,ANU,DEKK,I_APPLY_FORCE)
C
      IMPLICIT REAL*8(A-H,O-Z)
C
      DIMENSION RHSS(22),RV1(22)
      DIMENSION SDEV(4),BDEVTRAN(22,4),BV(22),ANPMATX(22)
C
      RV1 = MATMUL(BDEVTRAN,SDEV)

      IF (I_APPLY_FORCE.EQ.1) THEN
        RHSS = RHSS - DVOLU*( RV1 + PRESS*BV +
     +(DEKK - 3.D0*(1.D0-2.D0*ANU)*DPRESS/E)*ANPMATX )
      ELSE
        RHSS = RHSS + DVOLU*( RV1 + PRESS*BV +
     +(DEKK - 3.D0*(1.D0-2.D0*ANU)*DPRESS/E)*ANPMATX )
      END IF
C
      END
C
C*****************************************************************************
C
      SUBROUTINE KAMATRIX(AMATRX,DDSDDE,BDEV,BDEVTRAN,BV,ANPMATX,DVOLU,
     + E,ANU,NDOFEL,I_APPLY_FORCE)
C
      IMPLICIT REAL*8(A-H,O-Z)
C
      DIMENSION AMATRX(NDOFEL,NDOFEL),DDSDDE(4,4),BDEV(4,22),
     + BDEVTRAN(22,4),BV(22),ANPMATX(22)
      DIMENSION BV1(22,1),ANPMATX1(1,22),BV1T(1,22),ANPMATX1T(22,1)
      DIMENSION V1(4,22),V2(22,22),BTV1(22,22),BTV2(22,22),BTV3(22,22),
     + BTV4(22,22)
C
      V1 = MATMUL(DDSDDE,BDEV)
      BTV1 = MATMUL(BDEVTRAN,V1)
C
      BV1(1:NDOFEL,1) = BV(1:NDOFEL)
      ANPMATX1(1,1:NDOFEL) = ANPMATX(1:NDOFEL)
      BTV2 = MATMUL(BV1,ANPMATX1)
```

```
817 C
818       BV1T = TRANSPOSE(BV1)
819       ANPMATX1T = TRANSPOSE(ANPMATX1)
820       BTV3 = MATMUL(ANPMATX1T,BV1T)
821 C
822       DO I=1,NDOFEL
823       DO J=1,NDOFEL
824         V2(I,J) = ANPMATX(I)*ANPMATX(J)
825       END DO
826       END DO
827       BTV4 = (3.D0/E)*(1.D0 - 2.D0*ANU)*V2
828 C
829
830       IF (LAPPLY_FORCE.EQ.1) THEN
831         AMATRX = AMATRX + DVOLU*(BTV1 + BTV2 + BTV3 - BTV4)
832       ELSE
833         AMATRX = AMATRX - DVOLU*(BTV1 + BTV2 + BTV3 - BTV4)
834       END IF
835 C
836       END
837 C
838 C**********************************************************************
839 C
840       SUBROUTINE KUMAT(SDEV,STATEV,DDSDDE,TIMEE,DTIME,CMNAME,
841      + NDI,NSHR,NTENS,NSTATV,PROPSU,NPROPS,COORDS,DROT,PNEWDT,
842      + DFGRD0,DFGRD1,NOEL,NPT,KSTEP,KINC,IOUT,IWR,SIGY,DEKK)
843 C
844       IMPLICIT DOUBLE PRECISION(A-H,O-Z)
845       CHARACTER*80 CMNAME
846 C
847       DIMENSION SDEV(NTENS),STATEV(NSTATV),DDSDDE(NTENS,NTENS),
848      + TIMEE(2),PROPSU(NPROPS),COORDS(2),DROT(3,3),
849      + DFGRD0(3,3),DFGRD1(3,3)
850 C
851       DIMENSION R(3,3),DETENS(3,3)
852 C
853       DIMENSION QMX(4,4),AIMX(4,4),AJMX(4,4),
854      + AKMX(4,4),ELASTICD(4,4),DE(4),AN(4),DEV(4),SEL(4)
855 C
856       MTENS=4
857       IF (IWR /= 0) THEN
858         WRITE(IOUT,*)
859         WRITE(IOUT,*) ' ---- Starting UMAT ----'
860         WRITE(IOUT,*) 'KSTEP, KINC, TOTAL TIME'
861         WRITE(IOUT,1004) KSTEP,KINC,TIMEE(2)
862         WRITE(IOUT,*) 'NOEL, NPT, NTENS'
863         WRITE(IOUT,1002) NOEL,NPT,NTENS
864       END IF
865 C
866       IF (MTENS /= NTENS) THEN
867         WRITE(IOUT,*) ' MTENS /= NTENS PROGRAM STOPS IN UMAT. '
868         WRITE(IOUT,*) ' MTENS=',MTENS,' NTENS=',NTENS
869         WRITE(IOUT,*) ' ELEMENT ',NOEL,' NPT ',NPT
870         CALL XIT
871       END IF
872 C
```

```
873         AIMX = 0.D0 ! AIMX(I,J) = 0.
874         AJMX = 0.D0 ! AJMX(I,J) = 0.
875         DO I=1,NDI
876           AIMX(I,I) = 1.D0
877         END DO
878         DO I=NDI+1,NTENS
879           AIMX(I,I) = 0.5D0     ! I=(1/2)( ik jl+ il jk), 4x4 tensor
880         END DO
881 C
882         AJMX(1:NDI,1:NDI) = 1.D0/3.D0 !J=(1/3)     , 4x4 tensor
883 C
884         AKMX = AIMX - AJMX    ! AKMX(I,J)=AIMX(I,J)-AJMX(I,J), K=I-J
885 C
886         E    = PROPSU(1)
887         ANU  = PROPSU(2)
888         SIG0 = PROPSU(3)
889         EXPO = PROPSU(4)
890         G = E/(2.D0*(1.D0 + ANU))
891         E0 = SIG0/E
892         AUX_2G = 2.D0*G
893         ELASTICD = AUX_2G*AKMX  !L = 2G*K + 3  *J, 4x4 tensor
894 C
895         EBARN  = STATEV(1)
896         YFLAGN = STATEV(2)
897 C
898         AUX = SUM(DABS(DFGRD1-DFGRD0))
899 C
900         IF (IWR /= 0) THEN
901           WRITE(IOUT,*) 'Initial state variables: EBARN, YFLAGN'
902           WRITE(IOUT,1001) STATEV(1),STATEV(2)
903           WRITE(IOUT,*) 'DFGRD0'
904           DO I=1,3
905             WRITE(IOUT,1001) (DFGRD0(I,J),J=1,3)
906           ENDDO
907           WRITE(IOUT,*) 'DFGRD1'
908           DO I=1,3
909             WRITE(IOUT,1001) (DFGRD1(I,J),J=1,3)
910           ENDDO
911           WRITE(IOUT,*) 'SUM(DABS(DFGRD1-DFGRD0))'
912           WRITE(IOUT,1001) AUX
913         END IF
914 C
915 C
916         IF (AUX > 1.D-8) GOTO 29
917 C
918 ! ---------------------------------------------- DE = 0 ----------------------
919 C*** DE=0 NEEDS DDSDDE
920 C
921         DEKK = 0.D0
922         IF (YFLAGN == 0.D0) THEN
923 C
924 C*** ------------------ ELASTICITY ---
925           IF (IWR /= 0) THEN
926             WRITE(IOUT,*) 'DEMAG=0,  ELASTIC DDSDDE'
927           END IF
928           DDSDDE  = AUX_2G*AKMX
```

```
C
      ELSE !IF (YFLAGN /= 0.D0) THEN
C*** ------------------ PLASTICITY
        CALL KFINDN(SDEV,AN,NDI,NSHR,NTENS)
        CALL KSIGY(SIGYN,HN,SIG0,EXPO,E0,EBARN)
        AUXPL = 4.D0*G/(3.D0 + HN/G)
        DO I=1,NTENS
        DO J=1,NTENS
          DDSDDE(I,J) = AUX_2G*AKMX(I,J) - AUXPL*AN(I)*AN(J)
        ENDDO
        ENDDO
C
      END IF
C
      IF (IWR /= 0) THEN
        WRITE(IOUT,*) 'DDSDDE'
        DO I=1,NTENS
          WRITE(IOUT,1001) (DDSDDE(I,J),J=1,NTENS)
        ENDDO
        WRITE(IOUT,*) '----------------------'
        WRITE(IOUT,*) 'EXITS UMAT'
      END IF
C
      RETURN
C
  29  CONTINUE
C
! ------------------------------------------------------------ DE /= 0 ----------
!*** INTEGRATE ELASTOPLASTIC EQUATIONS
C
C*** FIND THE LOGARITHMIC STRAIN TENSOR
      CALL KELOGR(DFGRD0,DFGRD1,DETENS,R)
      CALL KTTOVSTRN(DETENS,DE,NDI,NSHR,NTENS)
      DEKK = SUM(DE(1:NDI))
C
      IF (IWR /= 0) THEN
        WRITE(IOUT,*) 'LOGARITHMIC DE'
        DO I=1,3
          WRITE(IOUT,1001) (DETENS(I,J),J=1,3)
        ENDDO
        WRITE(IOUT,*) 'ROTATION TENSOR R'
        DO I=1,3
          WRITE(IOUT,1001) (R(I,J),J=1,3)
        ENDDO
        WRITE(IOUT,*) 'DE VECTOR (SHEAR COMPONENTS are ENGNG)'
        WRITE(IOUT,1001) (DE(I),I=1,NTENS)
      END IF
C
C*** Find elastic predictor SEL
C
      DEV = MATMUL(AKMX,DE)
      SEL = SDEV + AUX_2G*DEV ! s(elastic)=sn+2G e
      CALL KINVAR(SEL,PEL,QEL,NDI,NSHR,NTENS)
      CALL KSIGY(SIGYN,HN,SIG0,EXPO,E0,EBARN)
      PHI = QEL - SIGYN
      IF (IWR /= 0) THEN
```

```
985           WRITE(IOUT,*)  ' SEL'
986           WRITE(IOUT,1001)  (SEL(I),I=1,NTENS)
987           WRITE(IOUT,*)  ' PEL, QEL'
988           WRITE(IOUT,1001)  PEL, QEL
989           WRITE(IOUT,*)  ' SIGYN, HN'
990           WRITE(IOUT,1001)  SIGYN,HN
991           WRITE(IOUT,*)  ' PHI'
992           WRITE(IOUT,1001)  PHI
993         END IF
994 C
995         IF  (PHI > 0.D0) GOTO 2000
996 C
997 C*** ---------------- ELASTICITY
998 C
999   1000 CONTINUE
1000        YFLAG = 0.D0
1001        SDEV = SEL     !SDEV(I) = SEL(I)
1002 C
1003 C***  ROTATE STRESS VECTOR AND UPDATE STATEV
1004 C
1005        CALL KROTSTRS(SDEV,R,QMX,NTENS)
1006        STATEV(2) = YFLAG
1007 C
1008 C*** ELASTIC JACOBIAN
1009 C
1010        DDSDDE = AUX_2G*AKMX
1011        SIGY = SIGYN
1012 C
1013        GOTO 9999
1014 C
1015 C*** ----------------  PLASTICITY
1016 C
1017   2000 CONTINUE
1018        YFLAG = 1.D0
1019 C
1020 C*** DETERMINE DEBAR
1021 C
1022        DEBAR = (QEL - SIGYN)/(3.D0*G)
1023        EBAR = EBARN + DEBAR
1024        IF  (IWR /= 0) THEN
1025          WRITE(IOUT,*)  'FIRST ESTIMATE FOR DEBAR, EBAR=EBARN+DEBAR'
1026          WRITE(IOUT,1001)  DEBAR, EBAR
1027        END IF
1028 C
1029        YTOL = SIG0*1.D-6
1030        DO ILOOP=1,20
1031          CALL KSIGY(SIGY,H,SIG0,EXPO,E0,EBAR)
1032          FCN=QEL-3.D0*G*DEBAR-SIGY  ! YIELD CONDITION
1033          IF  (DABS(FCN) <= YTOL) GOTO 2011
1034          DFJAC = -3.D0*G - H   ! DERIVATIVE
1035          DDE = -FCN/DFJAC
1036          DEBAR = DEBAR + DDE
1037          EBAR = EBARN + DEBAR
1038          IF  (EBAR < 0.D0) EBAR=0.D0
1039        END DO
1040        WRITE(IOUT,*)
```

```
1041        + 'NEWTON LOOP IN UMAT DOES NOT CONVERGE. PNEWDT=0.5'
1042          PNEWDT=0.5D0
1043          RETURN
1044  C
1045   2011   CONTINUE
1046  C
1047          IF (IWR /= 0) THEN
1048            WRITE(IOUT,*) 'Newton loop in UMAT converged'
1049            WRITE(IOUT,*) 'NUMBER OF NEWTON ITERATIONS'
1050            WRITE(IOUT,1002) ILOOP
1051            WRITE(IOUT,*) 'DEBAR,EBAR'
1052            WRITE(IOUT,1001) DEBAR,EBAR
1053          END IF
1054  C
1055          CALL KFINDN(SEL,AN,NDI,NSHR,NTENS)
1056          SDEV = SEL - AUX_2G*DEBAR*AN
1057          IF (IWR /= 0) THEN
1058            WRITE(IOUT,*) 'AN'
1059            WRITE(IOUT,1001) AN
1060            WRITE(IOUT,*) 'SDEV'
1061            WRITE(IOUT,1001) SDEV
1062          END IF
1063  C
1064  C***    ROTATE STRESS VECTOR AND NORMAL AN
1065          CALL KROTSTRS(SDEV,R,QMX,NTENS)
1066          CALL KROTSTRS(AN,R,QMX,NTENS)
1067          IF (IWR /= 0) THEN
1068            WRITE(IOUT,*) 'ROTATED SDEV'
1069            WRITE(IOUT,1001) SDEV
1070          END IF
1071  C
1072  C*** UPDATE STATE VARIABLES
1073          STATEV(1) = EBAR
1074          STATEV(2) = YFLAG
1075          IF (IWR /= 0) THEN
1076            WRITE(IOUT,*) 'Updated STATE VARIABLES: EBAR, YFLAG'
1077            WRITE(IOUT,1001) STATEV(1), STATEV(2)
1078          END IF
1079  C
1080  C
1081  C*** CALCULATE JACOBIAN
1082          AUXPL1 = 4.D0*G/(3.D0 + H/G)
1083          AUXPL2 = 4.D0*DEBAR*G*G/QEL    !QEL: s_equivalent elastic
1084          DO I=1,NTENS
1085          DO J=1,NTENS
1086            DDSDDE(I,J) = AUX_2G*AKMX(I,J) - 1.5D0*AUXPL2*AKMX(I,J) -
1087        +                 (AUXPL1-AUXPL2)*AN(I)*AN(J)
1088          ENDDO
1089          ENDDO
1090  C
1091   9999   CONTINUE
1092  C
1093          IF (IWR /= 0) THEN
1094            WRITE(IOUT,*) 'DDSDDE = '
1095            DO I=1,NTENS
1096              WRITE(IOUT,1001) (DDSDDE(I,J),J=1,NTENS)
```

```
1097            END DO
1098            WRITE(IOUT,*)  ' ---- Finished UMAT ----'
1099            WRITE(IOUT,*)
1100          END IF
1101  C
1102  C
1103     1001 FORMAT(1P8E13.5)
1104     1002 FORMAT(10I5)
1105     1003 FORMAT(I5,1P8E13.5)
1106     1004 FORMAT(2I5,1P7E13.5)
1107          END
1108  C
1109  C**********************************************************************
1110  C
1111          SUBROUTINE KROTSTRS(A,R,QMX,NTENS)
1112  C
1113          IMPLICIT REAL*8(A-H,O-Z)
1114  C
1115          DIMENSION A(NTENS),R(3,3),QMX(4,4),AUX(4)
1116  C
1117          QMX = 0.D0     ! QMX(I,J) = 0.
1118          DO I=1,3
1119          DO J=1,3
1120            QMX(I,J) = R(I,J)*R(I,J)
1121          END DO
1122          END DO
1123  C
1124          QMX(1,4) = 2.D0*R(1,1)*R(1,2)
1125          QMX(2,4) = 2.D0*R(2,1)*R(2,2)
1126          QMX(3,4) = 2.D0*R(3,1)*R(3,2)
1127  C
1128          QMX(4,1) = R(1,1)*R(2,1)
1129          QMX(4,2) = R(1,2)*R(2,2)
1130          QMX(4,3) = R(1,3)*R(2,3)
1131  C
1132          QMX(4,4) = R(1,2)*R(2,1) + R(2,2)*R(1,1)
1133  C
1134          AUX = MATMUL(QMX,A)
1135          A = AUX ! A(I) = AUX(I)
1136  C
1137          END
1138  C
1139  C**********************************************************************
1140  C
1141          SUBROUTINE KELOGR(DFGRD0,DFGRD1,DETENS,R)
1142  C
1143          IMPLICIT REAL*8(A-H,O-Z)
1144  C
1145          DIMENSION DFGRD0(3,3),DFGRD1(3,3),DETENS(3,3),R(3,3)
1146          DIMENSION DFGINV(3,3),DF(3,3),CC(3,3),WORK(6),PS(3),ANN(3,3),
1147         + UINV(3,3),DFT(3,3)
1148  C
1149          CALL KINV3X3(DFGRD0,DFGINV)
1150          DF = MATMUL(DFGRD1,DFGINV)
1151  C
1152  C*** EXACT CALCULATION OF LOGARITHMIC STRAIN
```

```
1153 C
1154       DFT = TRANSPOSE(DF)
1155       CC = MATMUL(DFT,DF)
1156       WORK(1) = CC(1,1)
1157       WORK(2) = CC(2,2)
1158       WORK(3) = CC(3,3)
1159       WORK(4) = CC(1,2)
1160       WORK(5) = CC(1,3)
1161       WORK(6) = CC(2,3)
1162       CALL SPRIND(WORK,PS,ANN,1,3,3) !Find principal values & directions
1163 C
1164       CALL KARRANGE(PS,ANN)
1165       DO I=1,3
1166         PS(I) = DSQRT(PS(I))
1167       END DO
1168 C
1169       DO I=1,3
1170       DO J=1,3
1171         DETENS(I,J) =  DLOG(PS(1))*ANN(1,I)*ANN(1,J)
1172      +                + DLOG(PS(2))*ANN(2,I)*ANN(2,J)
1173      +                + DLOG(PS(3))*ANN(3,I)*ANN(3,J)
1174         UINV(I,J)   =   (1.D0/PS(1))*ANN(1,I)*ANN(1,J)
1175      +                + (1.D0/PS(2))*ANN(2,I)*ANN(2,J)
1176      +                + (1.D0/PS(3))*ANN(3,I)*ANN(3,J)
1177       END DO
1178       END DO
1179 C
1180       R = MATMUL(DF,UINV)
1181 C
1182       END
1183 C
1184 C**********************************************************************
1185 C
1186       SUBROUTINE KFINDN(S,AN,NDI,NSHR,NTENS)
1187 C
1188 C*** Given the stress tensor S, find the normal to the yield surface AN
1189 C
1190       IMPLICIT REAL*8(A-H,O-Z)
1191       DIMENSION S(NTENS),AN(NTENS),SDEV(6)
1192 C
1193       CALL KINVAR(S,P,Q,NDI,NSHR,NTENS)
1194 C
1195       SDEV(1:NTENS) = S(1:NTENS)
1196       SDEV(1:NDI) = SDEV(1:NDI) - P
1197 C
1198       AUX = 1.5D0/Q
1199       AN(1:NTENS) = AUX*SDEV(1:NTENS)   ! AN(I) = AUX* SDEV(I)
1200 C
1201       END
1202 C
1203 C**********************************************************************
1204 C
1205       SUBROUTINE KARRANGE(PS,ANN)
1206 C
1207       IMPLICIT REAL*8(A-H,O-Z)
1208 C
```

```
1209        DIMENSION PS(3),ANN(3,3),PSC(3),ANNC(3,3)
1210  C
1211  C*** LABELS EIGENVALUES SO THAT PS(1)>PS(2)>PS(3) AND
1212  C*** REARRANGES EIGENVECTORS ACCORDINGLY
1213  C
1214        PSC = PS    ! PSC(I) = PS(I)
1215        ANNC = ANN ! ANNC(I,J) = ANN(I,J)
1216  C
1217        IF (PS(1) >= PS(2) .AND. PS(2) >= PS(3)) THEN
1218          IMAX = 1
1219          IINT = 2
1220          IMIN = 3
1221          GOTO 999
1222        END IF
1223  C
1224        IF (PS(1) >= PS(3) .AND. PS(3) >= PS(2)) THEN
1225          IMAX = 1
1226          IINT = 3
1227          IMIN = 2
1228          GOTO 999
1229        END IF
1230  C
1231        IF (PS(2) >= PS(1) .AND. PS(1) >= PS(3)) THEN
1232          IMAX = 2
1233          IINT = 1
1234          IMIN = 3
1235          GOTO 999
1236        END IF
1237  C
1238        IF (PS(2) >= PS(3) .AND. PS(3) >= PS(1)) THEN
1239          IMAX = 2
1240          IINT = 3
1241          IMIN = 1
1242          GOTO 999
1243        END IF
1244  C
1245        IF (PS(3) >= PS(1) .AND. PS(1) >= PS(2)) THEN
1246          IMAX = 3
1247          IINT = 1
1248          IMIN = 2
1249          GOTO 999
1250        END IF
1251  C
1252        IF (PS(3) >= PS(2) .AND. PS(2) >= PS(1)) THEN
1253          IMAX = 3
1254          IINT = 2
1255          IMIN = 1
1256          GOTO 999
1257        END IF
1258  C
1259   999  CONTINUE
1260        PS(1) = PSC(IMAX)
1261        PS(2) = PSC(IINT)
1262        PS(3) = PSC(IMIN)
1263        DO I=1,3
1264          ANN(1,I) = ANNC(IMAX,I)
```

```
1265            ANN(2,I) = ANNC(IINT,I)
1266          END DO
1267  C
1268  C*** THE THIRD UNIT EIGENVECTOR IS THE CROSS PRODUCT OF THE FIRST TWO
1269  C*** SO THAT THE UNIT EIGENVECTORS FORM A RIGHT-HANDED BASE
1270          ANN(3,1) =     ANN(1,2)*ANN(2,3) - ANN(2,2)*ANN(1,3)
1271          ANN(3,2) = -( ANN(1,1)*ANN(2,3) - ANN(2,1)*ANN(1,3) )
1272          ANN(3,3) =     ANN(1,1)*ANN(2,2) - ANN(2,1)*ANN(1,2)
1273  C
1274          END
1275  C
1276  C**********************************************************************
1277  C
1278          SUBROUTINE KINVAR(STRESS,P,Q,NDI,NSHR,NTENS)
1279  C
1280  C Given the stress tensor, find the hydrostatic stress and
1281  C                          the von Mises equivalent stress
1282  C
1283          IMPLICIT REAL*8(A-H,O-Z)
1284  C
1285          DIMENSION STRESS(NTENS),SDEV(6)
1286  C
1287          P = SUM(STRESS(1:NDI))/3.D0
1288  C
1289          SDEV = 0.D0
1290          SDEV(1:NTENS) = STRESS(1:NTENS) ! SDEV(I) = STRESS(I)
1291          SDEV(1:NDI) = SDEV(1:NDI) - P
1292  C
1293          AUX=DOT_PRODUCT(SDEV,SDEV)
1294          DO I=NDI+1,NTENS
1295            AUX = AUX + SDEV(I)*SDEV(I)
1296          END DO
1297          Q=DSQRT(1.5D0*AUX)
1298  C
1299          END
1300  C
1301  C**********************************************************************
1302  C
1303          SUBROUTINE KTTOVSTRN(ET,EV,NDI,NSHR,NTENS)
1304  C
1305          IMPLICIT REAL*8(A-H,O-Z)
1306  C
1307  C*** Tensor TO Vector STRainN
1308  C*** forms "vector" (EV) from tensor (ET) strain tensor
1309  C
1310          DIMENSION ET(3,3),EV(NTENS)
1311  C
1312          DO I=1,NDI
1313            EV(I) = ET(I,I)
1314          END DO
1315          DO I=1,NSHR
1316            IF (I == 1) EV(NDI+I) = 2.D0*ET(1,2)
1317            IF (I == 2) EV(NDI+I) = 2.D0*ET(1,3)
1318            IF (I == 3) EV(NDI+I) = 2.D0*ET(2,3)
1319          END DO
1320  C
```

```
1321       END
1322 C
1323 C**********************************************************************
1324 C
1325       SUBROUTINE KSIGY(SIGY,H,SIG0,EXPO,E0,EBAR)
1326 C
1327       IMPLICIT REAL*8(A-H,O-Z)
1328 C
1329 C*** Determine flow stress SIGY and slope H
1330 C
1331       IF (EXPO < 50.D0) THEN
1332         SIGY = SIG0*(1.D0+EBAR/E0)**(1.D0/EXPO)
1333         H = SIGY/(EXPO*(E0+EBAR))
1334       ENDIF
1335 C
1336       IF (EXPO >= 50.D0) THEN
1337         SIGY = SIG0
1338         H = 0.D0
1339       ENDIF
1340 C
1341       END
1342 C
1343 C**********************************************************************
1344 C
1345       SUBROUTINE UMAT(STRESS,STATEV,DDSDDE,SSE,SPD,SCD,
1346      + RPL,DDSDDT,DRPLDE,DRPLDT,
1347      + STRAN,DSTRAN,TIMEE,DTIME,TEMP,DTEMP,PREDEF,DPRED,CMNAME,
1348      + NDI,NSHR,NTENS,NSTATV,PROPS,NPROPS,COORDS,DROT,PNEWDT,
1349      + CELENT,DFGRD0,DFGRD1,NOEL,NPT,LAYER,KSPT,KSTEP,KINC)
1350 C
1351       INCLUDE 'ABA_PARAM.INC'
1352       CHARACTER*80 CMNAME
1353 C
1354       DIMENSION STRESS(NTENS),STATEV(NSTATV),
1355      + DDSDDE(NTENS,NTENS),DDSDDT(NTENS),DRPLDE(NTENS),
1356      + STRAN(NTENS),DSTRAN(NTENS),TIMEE(2),PREDEF(1),DPRED(1),
1357      + PROPS(NPROPS),COORDS(3),DROT(3,3),DFGRD0(3,3),DFGRD1(3,3)
1358 C
1359       STRESS = 0.D0
1360       DDSDDE = 0.D0
1361 C
1362       END
1363 C
1364 C**********************************************************************
1365 C
1366       SUBROUTINE UVARM(UVAR,DIRECT,T,TIME,DTIME,CMNAME,ORNAME,
1367      + NUVARM,NOEL,NPT,LAYER,KSPT,KSTEP,KINC,NDI,NSHR,COORD,
1368      + JMAC,JMATYP,MATLAYO,LACCFLA)
1369 C
1370       INCLUDE 'ABA_PARAM.INC'
1371 C
1372       CHARACTER*80 CMNAME,ORNAME
1373       DIMENSION UVAR(NUVARM),DIRECT(3,3),T(3,3),TIME(2)
1374       DIMENSION JMAC(*),JMATYP(*),COORD(*)
1375 C
1376       DIMENSION ZXEN(2000,9,20)
```

```
1377          COMMON/KNICK/ZXEN
1378  C
1379          NODEADD=1000
1380          NPLOT=12
1381          DO  I=1,NPLOT
1382             UVAR(I)=ZXEN(NOEL-NODEADD,NPT,I)
1383          ENDDO
1384  C
1385          END
1386  C
1387  C**********************************************************************
1388  C
```